Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Prispôsobiteľný Widget

(dokumentácia k 3. 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

Tretí šprint

Tretí šprint začínal 4. novembra 2010 a jeho koniec bol stanovený na 15. novembra 2010. Tento šprint bol o niečo kratší, pretože sa v týždni sa odovzdávanie výsledku predošlého šprintu posunulo o dva dni z dôvodu sviatku a rektorského voľna. V tomto šprinte sme si dali aj vzhľadom na jeho dĺžku za úlohu spojenie jednotlivých kódov funkcionality widgetu do jedného celku a to tak, aby bol výsledný kód v štruktúrovaný podľa zvyklostí použitého frameworku.

„User story“

Zadávateľ definoval požiadavku prepojenia štýlov s vygenerovaným widgetom, teda aby sa po zvolení vzhľadu widgetu navolený štýl prejavil vo výslednom vzhľade widgetu, keď ho používateľ vloží na používateľskú stránku. Ďalšou požiadavkou bolo, aby sa widget generoval spolu s nadefinovaným štýlom pri kustomizácii widgetu. Zadávateľ dal takisto požiadavku, aby sa všetky výstupy zjednotili do jednej webovej prezentácie. Vzhľadom na doterajšie nepoužitie žiadneho frameworku na kódovanie vedúci tímu odporučil použitie Zend frameworku, a preto je jednou z požiadaviek takisto preklopiť celý kód do tohto frameworku. Nadväzujúcou požiadavkou bolo preskúmanie použitia restfull API pre náš projekt s využitím Zend frameworku. Vedúci projektu takisto dal požiadavku, aby bolo využívané SVN na prácu s kódom a nie práca na lokálnych serveroch.

Analýza

Analýza frameworkov

Zend Framework

Na odporúčanie vedúceho projektu sme sa najskôr rozhodli pre kódovanie vo frameworku Zend. Zend je open-source framework určený pre vývoj webových aplikácií a služieb v PHP5. Zend je implementovaný použitím 100% objektovo orientovaného kódu. Komponentová štruktúra Zend frameworku je jedinečná, pričom každý component je navrhnutý s istou iba malou náväznosťou na iné komponenty. Takáto voľne previazaná architektúra poskytuje

vývojárom možnosť používať tieto komponenty jednotlivo. Napriek tomu, že je možné ich používať oddelene, kombináciou komponentov štandardnej knižnice Zend frameworku je možné dosiahnuť silný a rozšíriteľný webový framework pre webovú aplikáciu. Zend framework ponúka jednoducho použiteľnú, robustnú, vysoko výkonnú MVC(model-control-view) implementáciu, databázovú abstrakciu a komponent form-ov, ktoré implementujú HTML renderovanie, validáciu a filtrovanie tak, aby vývojári mohli konsolidovať všetky tieto operácie použitím jedného ľahko ovládateľného, objektovo-orientovaného rozhrania. Komponenty ako Zend_Auth a Zend_Acl poskytujú používateľskú autentifikáciu a autorizáciu podľa všetkých doporučení. Ostatné komponenty implementujú klientské knižnice na jednoduchý prístup k najpopulárnejším dostupným webovým službám. Čokoľvek takáto aplikácia potrebuje, Zend framework poskytuje komponent, ktorý môže byť použitý na významné zníženie času potrebného na vývoj s poskytnutým testovacím podkladom. Zend framework má významných partnerov ako Google, Microsoft, ktorí Zend frameworku poskytujú rozhrania pre webové služby a iné technológie, ktoré sú ochotní sprístupniť vývojárom používajúcim Zend framework.

Po diskusii s konzultantom pre náš projekt sme usú dili a aj na jeho odporúčanie sme sa rozhodli, že použitie Zend frameworku je pre náš prjekt zbytočné, pretože je to príliš robustný nástroj a poskytuje obrovské množstvo funkcionality, ktorú my nevyužijeme. Preto sme sa v druhej polovici tohto šprintu rozhodli pre iný, jednoduchší framework na základe analýzy a odporúčaní konzultanta.

Nette Framework

- MVC/MVP podpora
- Sústredí sa aj na bezpečnosť
- Podporuje AJAX, s ktorým pracujeme
- JQuery
- Využíva objektový prístup v rámci PHP 5
- Možnosť prepojenia s inými frameworkmi a ich vzájomné kombinovanie (napr. aj so ZEND)
- Podporuje prácu so šablónami
- Jednoduchá inštalácia
- Veľmi dobrý výkon

- Jednoduchá práca s databázou (MySQL). Funkcionalita je však trochu chudobná, no pre náš projekt postačujúca
- Podporuje znovupoužiteľnosť kódu
- Prehľadne spracovaný návod

Cake php

- Podporuje MVC
- Nevyžaduje ďalšie konfigurácie, ani pri napojení na databázu
- Sústredí sa na bezpečnosť, zahŕňa autentifikáciu, prácu so session ...
- Podporuje objektovo orientovaný prístup

Oba frameworky sú si podobné. Cake php poskytuje o niečo lepšie možnosti, napr. pri práci s databázou. Nette je však pre náš vývoj postačujúci. Keďže je o niečo menší je ľahšie zvládnuteľný. V prípade ak by sme počas vývoja natrafili na problém, ktorý nie je jednoducho riešiteľný prostredníctvom Nette, poskytuje nám možnosť pridania externých knižníc z iných frameworkov ako napríklad ZEND. Framework Nette sa používa pri vývoji webových aplikácií malého až stredného rozsahu v jazyku PHP. Je navrhnutý tak, aby podporoval bezpečnosť aplikácie (validácie formulárov, adresárová štruktúra...), čím šetrí čas pri vývoji. Výkonnostne je na veľmi dobrej úrovni v porovnaní s frameworkami ako je napríklad ZEND dosahuje lepšie rýchlostné výsledky. Framework Nette je aktívne vyvíjaný v Českej republike práve českými programátormi a má teda dobrú podporu, čo sa týka aktualizácií. Nevýhoda je v slabšej dokumentácii a nejasných návodoch, ktoré sú nie vždy aktuálne pre danú verziu. Našťastie je základňa používateľov tohto frameworku pomerne rozsiahla, a preto sa podpora dá nájsť u používateľov samotných.

Návrh

Nette podporuje vývoj aplikácie pod návrhovým vzorom MVC (model-view-controler) model. MVC model je v Nette upravený, prispôsobený a použiteľný pod názvov MVP (model-view-presenter). Vytvoríme stránku vo frameworoku Nette, ktorá bude slúžiť na vytvorenie a prezentáciu Widgetu. Používateľ si tak bude môcť navoliť svoj vlastný Widget, prečítať si návod na prácu s widgetom alebo sa o ňom dozvedieť niečo viac priamo z našej stránky. Všetky potrebné informácie, spolu s možnosťou vygenerovania widgetu, budú

k dispozícii na jednom mieste. Funkcionalita samotného widgetu sa z pohľadu používateľa meniť nebude, ale bude preprogramovaná do Nette frameworku.

Implementácia

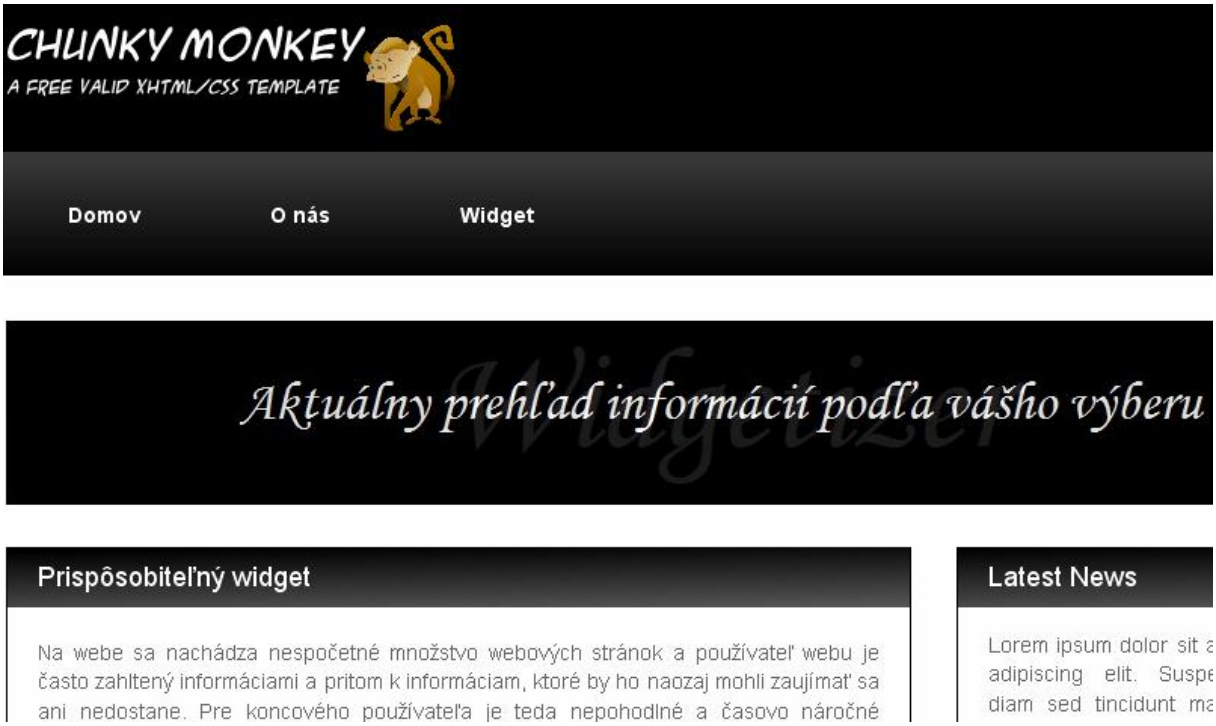
Webová stránka pre vytvorenie a prezentáciu widgetu

Na prezentáciu nášho projektu sme vytvorili webovú stránku, ktorá bude poskytovať základné informácie o projekte, o členoch tímu, ale aj informácie o widgete načo slúži, aké sú výhody jeho použitia a aj samotný návod ako si vlastný widget vytvoriť. Ako základný stavebný kameň pre stránku sme použili voľne dostupnú šablónu, ktorú sme stiahli z web stránky www.freewebtemplates.com. Šablóna, ktorú sme použili obsahovala základný xhtml validný kód a css štýly. Avšak aby sme dodržali adresárovú štruktúru frameworku Nette, bolo potrebné zmeniť štruktúru šablóny podľa pravidiel tohto frameworku. Táto štruktúra nie je povinná, ale rozhodli sme sa ju používať. Všetky súbory .css so štýlmi, obrázky a súbory typu javascript bolo nutné oddeliť od html súborov a php súborov. Všetky boli umiestnené podľa konvencií frameworku Nette do adresára /document_root a následne do jednotlivých zložiek:

- Štýly boli umiestnené do adresára /css
- Obrázky do /images
- Skripty do /js

Hlavná štruktúra stránky sa presunula do adresára /app, ktorý je centrom všetkej aplikačnej logiky. Aby sa dodržala MVP konvencia frameworku Nette vytvorili sme adresáre /models, /presenters a /templates. V adresári templates sú uložené šablóny, ktoré tvoria výzor aplikácie. Sú to v podstate kúsky html kódov. V tomto adresári je uložený takisto súbor @layout.phtml, ktorý obsahuje základný layout stránky ako hlavička, menu, pätička stránky. Do tohto layoutu budú vkladané jednotlivé šablóny podľa potreby. V adresári presenters sú súbory .php, ktoré riadia aplikáciu a vyvolávajú rôzne akcie. V modeli budú súbory, ktoré budú riadiť prácu aplikácie s dátami a databázou. Budú vyberať dáta z databázy a poskytovať ich presenterom.

Obrázok č.1: Zobrazenie výzoru našej stránky



Preprogramovanie časti funkcionality do Nette

Po prekonvertovaní šablóny do frameworku, bolo takisto prerobiť aj už existujúcu funkcionalitu widgetu. Bolo potrebné oddeliť zobrazovacie šablóny predstavujúce časť View z MVP modelu a jednotlivé funkcie, riadiace zobrazenie daných šablón predstavujúce Presenter z MVP modelu. Hlavná funkcionalita widgetu bola teda umiestnená v adresári */app/presenters* v súbore *WidgetPresenter.php*. Funkcie v tomto súbore následne volajú html šablóny zobrazujúce funkcionalitu widgetu. Tieto sme umiestnili do adresára /app/templates/Widget. Pre prácu s dátami ako sa v MVP modeli využíva Model. Preto sme jednotlivé funkcie, ktoré slúžia na prácu s dátami, parsovanie kategórií dátových zdrojov, získavanie dát z google spreadsheet umiestnili do súboru *WidgetModel.php*. Ten sme uložili do adresára */app/models*. Nasledujúci zdrojový kód predstavuje ukážku funkcionality nášho widgetu. Je to funkcia *processFilterForm* z presenteru *WidgetPresenter.php*. Jej úloha je riadiť volanie šablón, po odoslaní formulára na výber dátového zdroja čiže rss alebo zvolení kategórie z google spreadsheet.

```
public function processFilterForm(NAppForm $filterForm)
{
      if($filterForm['send']->isSubmittedBy())
      {
```

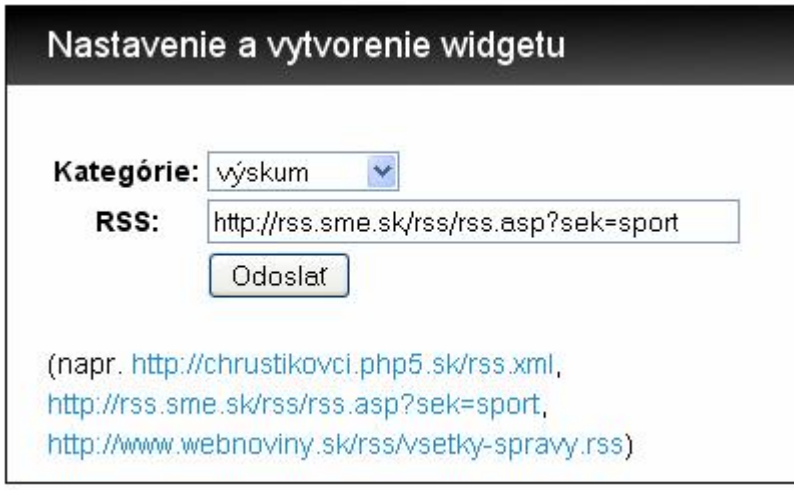
```
            $values = $filterForm->getValues();
            if(strlen($values['rss'])==0)
            {
                  $pom=$values['category'];
                  $category=$this->categories[$pom];
            }
            else
                  $category=-1;
      }
      if($category>-1)
            $this->redirect('Widget:createStyle',$category);

      else
            $this->redirect('Widget:showFilters',$values['rss'],$category);

}
```

Na nasledujúcom obrázku je zobrazený spôsob výberu dátového zdroja. Používateľ si zvolí kategóriu podujatia z google spredheet, ktorú chce zobrazovať vo svojom widgete alebo si zvolí nejaký iný zdroj dát, ktorý však musí byť vo formáte RSS. Po odoslaní formulára tlačidlom Odoslať sa zavolá hore spomenutá funkcia, ktorá spracuje zadané dáta z formulára a na základe nich sa rozhodne čo sa následne zobrazí.

Obrázok č.2: Zobrazenie výberu dátového zdroja

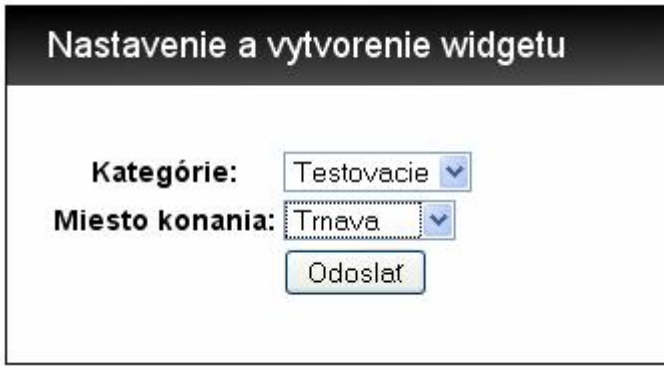


Výber dátového zdroja

Ako dátový zdroj pre widget sme umožnili používali dva typy súborov. Prvý je google spreadsheet, ktorý nám poskytol product owner a obsahuje dáta v podobe podujatí, ktoré majú rôzne kategórie. Ako druhý spôsob sme poskytli používateľom možnosť definovať si externý dátový zdroj vo forme rss súboru. Pri prvom spôsobe sme filtrovali dáta z tohto zdroja

a získali sme si všetky kategórie. Tieto sú poskytnuté používateľovi vo forme combo boxu. Po výbere kategórie sa mu budú vo widgete zobrazovať údaje iba danej kategórie.
Pri zadaní externého rss zdroja dát a odoslaní formulára sa daný zdroj dát spracuje aby sa z neho získali kategórie. Tieto sa následne zobrazia na ďalšej stránke a po zvolení kategórie sa budú používateľovi vo vytvorenom widgete zobrazovať iba tieto dáta. Uvažovali sme aj situáciu, že xml súbor s rss dátami bude obsahovať aj ďalšie informácie, podľa ktorých by bolo možné dáta filtrovať ako napríklad miesto konania podujatia. V prípade takéhoto súboru sa získajú tieto dodatočné informácie a zobrazia sa používateľovi na filtráciu. Informácie, ktoré sa budú zobrazovať v ním definovanom widgete, budú spĺňať jeho voľbu rozšíreného filtrovania.
Nasledujúci obrázok zobrazuje situáciu, keď používateľ zadá zdroj s rss dátami, obsahujúcimi aj údaje o mieste konania podujatia. Používateľovi sa zobrazí formulár s rozšíreným filtrovaním údajov.

Obrázok č.3: Zobrazenie formulára s rozšíreným filtrovaním údajov



Štýl vzhľadu

Z implementačného hľadiska je štýl Widgetu vytváraný dynamicky s použitím technológií HTML DOM, XHTML, CSS a JavaScript. S pomocou jazyka XHTML sú vytvorené jednotlivé časti „okna“ widgetu, teda tej časti aplikácie, ktorá je viditeľná bežnému používateľovi (návštevníkovi stránky na ktorej sa widget nachádza). Vďaka tomu, že jednotlivé časti (elementy) sú pomenované, je možné sa na ne dynamicky odvolávať a meniť ich vlastnosti.

Pri volaní elementov sa využíva Javascript, ktorý s využitím DOM štruktúry volá jednotlivé elementy widgetu a dynamicky, na základe používateľových preferencií a voľby mení parametre pomocou kaskádových štýlov. Týmto spôsobom je možné vytvárať dynamický náhľad toho, ako bude výsledný widget vyzerať na stránke.
V súčasnej verzii má osoba navrhujúca si widget na výber z nasledujúcich možností:

- názov okna widgetu
- možnosť dynamicky nastaviť šírku a výšku okna widgetu
- možnosť definovať maximálny počet článkov, ktoré sa budú v okne widgetu zobrazovať
- vybrať si spomedzi 14 farebných a tematických možností vzhľadu widgetu

Tieto parametre sa týkajú len samotného vzhľadu widgetu. Parametre, ktoré ovplyvňujú obsah okna widgetu sú definované v predchádzajúcom kroku pri voľbe zdroja dát. V tejto časti teda hovoríme len o tom, ako bude widget vyzerať – nie čo bude zobrazovať.
Z programovej stránky obsluhuje nastavovanie vzhľadu widgetu séria javascriptových funkcií uložených na servery v príslušnom priečinku s ostatnými súbormi tohto typu. Príslušná JS funkcia je volaná pomocou akcií ako onKeyUp, onChange a podobne, pričom toto volanie je súčasťou formulára a prebieha z pohľadu človeka okamžite, teda v reálnom čase.
Hodnoty z formulára sú odosielané na spracovanie ďalšej PHP funkcii, ktoré zo získaných hodnôt zostavia výsledný kód, ktorý si potom používateľ (správca nejakej stránky) môže vložiť do kódu svojej stránky.

Testovanie

V tomto šprinte nastala neštandardná situácia a vyskytli sa problémy s výstupom našej práce z dôvodu prechodu na ZEND framework. Na tomto frameworku sme sa dohodli na 1. stretnutí v tomto šprinte a začali sme s analýzou, ako korektne pracovať v ZEND frameworku a správne riešiť implementáciu úloh vzhľadom na využitie tohto frameworku. Na 2. stretnutí v tomto šprinte sme riešili problémy s implementáciou nášho existujúceho riešenia do spomenutého frameworku a pokračovali sme v preklápaní kódu do ZEND-u. Deň po tomto stretnutí sme mali diskusiu s externým konzultantom nášho projektu a po tejto diskusii sme uvážili, že bude potrebné framework zmeniť, pretože ZEND je pre náš projekt zbytočne robustný. Po analýze sme sa rozhodli implementovať náš kód v Nette frameworku, a tak sa zopakoval proces analýzy a implementácie vzhľadom na Nette framework. Keď sme však dokončili analýzu a začali implementovať, bol už takmer koniec šprintu a na vyhodnotení šprintu sme nemali ešte kód preklopený do Nette frameworku, aplikácia preto nemala ucelenú funkcionalitu a nebolo možné vykonať testovanie. Túto časť sme dorobili na začiatku 4. šprintu a výsledný efekt je rovnaký, ako v druhom šprinte, až na to, že Widgetizér už je súčasťou prehľadnej webovej stránky.