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

Prispôsobiteľný Widget

(dokumentácia k 4. 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

Štvrtý šprint

Štvrtý šprint začínal 15. novembra 2010 a jeho koniec bol stanovený na 29. novembra 2010. V tomto šprinte sme si dali za cieľ hlavne dokončiť preklopenie kódu do Nette frameworku a aktualizáciu webovej stránky a obsahu widgetu. Ďalej sme riešili využitie restful API a databázy v našom projekte.

„User story“

Zadávateľ zadal požiadavku definovať kód do Nette, doplniť výber kategórií v Nette a spracovať RSS pre rôzne stránky. Medzi ďalšie požiadavky patrí odstránenie výnimky na serveri, dať RSS z fóra do školskej stránky v podobe widgetu, ktorý bude generovaný widgetizérom a dať linku z fóra do školskej stránky. Ďalej bolo potrebné naštudovať ako fungujú REST služby, sprístupniť jednoduchú testovaciu metódu ako REST službu na školskom serveri, dopytovať externú REST službu: Alchemy API(poslať string, vrátiť kategóriu) a analyzovať iné externé služby použiteľné v rámci widgetu. Stanovená bola i požiadavka navrhnúť, ako bude komunikovať web aplikácia v nette s databázou, navrhnúť celkový databázovy model pre widgetizér, a keď používateľ bude chcieť vygenerovať widget, môže zadať svoju email adresu. Táto adresa sa uloží do tabuľky v db spolu s vygenerovaným url pre widget. Poslednou požiadavkou bolo rozšíriť parser, ktorý aktuálne spracúva len nadpis (title), a keďže widget obsahuje aktuálne len položku názov, tento názov musí byť hyperlinka, ktorá odkazuje na pôvodny zdroj.

Analýza

Vygenerovanie kódu

Po zvolení požadovaných hodnôt vo filtroch pri generovaní widgetu a následnom výbere štýlu je nutné na základe týchto hodnôt vygenerovať používateľovi kód, ktorý si uloží na svoju stránku a widget sa mu zobrazí. Vo widgete budú informácie, ktoré si používateľ vyfiltroval.

REST API

V dnešnej dobe sa stali REST APIs(REpresentational State Transfer Application Programming Interfaces) jedným z veľmi populárnych spôsobov programovania v oblasti webu. Je to hlavne z dôvodu, že používanie REST API je neuveriteľne jednoduché v akomkoľvek jazyku. Takisto je jednoduché vytvorenie takéhoto API, keďže v podstate nie je potrebné použiť nič iné ako špecifikáciu HTTP, ktorá je známa už dlhé obdobie. Pre každé API je prirodzené prijímanie požiadaviek. V REST API je typické definovanie URL schémy. Pri poskytovaní takejto služby je obvyklé mať štruktúru URL v podobe „/api/služba“, pričom typ služby závisí od operácie nad naším API.

Z dátového pohľadu dnes REST API najčastejšie pracujú nad štruktúrou XML alebo JSON. Tieto dve štruktúry sú najčastejšie používané pre prácu s údajmi poskytovanej služby. XML je široko rozšírený štandard, a preto je jeho používanie v dnešnej dobe úplne bežné a nie je problém pracovať nad takýmito údajmi. Podobne je už však na tom aj JSON, s ktorým sa jednoducho narába v jazyku JavaScript a jazyk PHP má jednoduché funkcie na zakódovanie a dekódovanie takéhoto formátu údajov. Robustnejšie REST APIs pracujú nad oboma množinami údajov, pričom špecifikácia použitej štruktúry sa často uvádza pri volaní služby vo vyskladanej URL.

Implementácia REST API je založená na použití http požiadaviek, ktoré pokrývajú požadovanú funkcionalitu REST služby. Je možné dokonca prirovnať jednotlivé požiadavky k štýlu CRUD(Create, Read, Update, Delete – teda vytvor, (na)čítaj, aktualizuj, vymaž) a to nasledujúcim spôsobom: GET = načítaj/získaj, POST = vytvor, PUT = aktualizuj, DELETE = vymaž. REST API teda samé rieši mnoho hlavných problémov vytvorenia nášho vlastného API, a to pomocou jednoduchých, ľahko pochopiteľných štandardov a protokolov.
REST teda rieši veľmi jednoducho požiadavky, no podobne jednoducho rieši i generovanie odpovedí. V REST architektúre odpovede sú obvykle dva hlavné komponenty a to telo odpovede a informačná časť(niekde iba status kód). S telom odpovede je jednoduché narábať, keďže ide o podobný princíp, ako v prípade požiadavky, to znamená použitie JSON alebo XML.

Pre tento projekt bola zvážená možnosť využitia vlastného, vytvoreného REST API, avšak v tejto fáze projektu sa takáto možnosť javila ako zbytočná, keďže bolo potrebné vytvoriť riadnu funkcionalitu samotného widget a nenašiel sa žiadny dôvod alebo spôsob, prečo by bolo potrebné vlastnú službu implementovať.

Na obohatenie funkcionality widgetu sme sa však rozhodli použiť API, ktoré poskytuje organizácia Alchemy. Toto API je dostupné na http://www.alchemyapi.com/ a poskytuje služby ako extrakcia entít z textu, kategorizácia textu, detekcia jazyka, označovanie konceptu, extrakcia kľúčových slov a ešte ďalšie služby, ktorých znalosť nie je pre tento projekt dôležitá. V tomto projekte sme sa rozhodli využiť službu kategorizácie textu, ktorá na základe zadaného textu a interného algoritmu vyhodnocovania kategórie určí kategóriu na základe kľúčových slov textu. Pre kategorizáciu sme sa rozhodli z dôvodu potreby kategorizácie udalostí, ktoré má náš widget zobrazovať. Táto služba dokáže určiť 12 kategórií a to: Arts&Entertainment, Business, Computer&Internet, Culture&Politics, Gaming, Health, Law&Crime, Religion, Recreation, Science&Technology, Sports a Weather. Služba má multilinguálnu podporu pre cudzie jazyky a dokáže kategorizovať text napísaný v 8 jazykoch a to: angličtina, francúzština, nemčina, taliančina, portugalčina, ruština, španielčina a švédčina.

Práca s dátami je možná v 4 formátoch, najpoužívanejšie sú dva spomínané JSON a XML, ostatné dva sú RDF a mikroformát(rel-tag). Príklad odpovede vo forme JSON:

```
{
   "status":"REQUEST_STATUS",
   "url":"DOCUMENT_URL",
   "category":"DETECTED_CATEGORY",
   "score":"CATEGORY_SCORE"
}
```

Kategorizácia môže byť vykonaná nad 3 rôznymi štruktúrami textu a to: voľný text, obsah URL adresy a obsah HTML stránky. V tejto fáze projektu sme využili možnosť kategorizovať na základe voľného textu, pričom ale zmena tohto vstupu je veľmi jednoduchá, takže boli odskúšané aj ostatné formáty vstupu.

Návrh

Štýl widgetu

Pred samotným prenosom funkcionality bolo potrebné naštudovať formálnu stránku vytvárania a práce s formulármi vo frameworku Nette. Našťastie tento framework podporuje veľmi dobre tvorbu frameworkov, uľahčuje mnohé úkony a ošetruje sám mnohé podmienky pre vstupné údaje do jednotlivých polí formulárov.

Na základe toho bolo pomerne ľahké navrhnúť jednoduchý formulár, ktorý z predošlého šprintu prenášal nasledujúcu funkcionalitu do frameworku Nette:

- názov widgetu
- počet článkov, ktoré sa budú zobrazovať
- rozmery widgetu (v rozmedzí pre šírku a výšku)
- typ farebného štýlu

Keďže framework Nette pracuje trochu iným štýlom s rozvrhnutím stránky a rozdeľuje ju do viacerých nezávislých celkov ktorými sú Model, Viewer a Presenter, bolo nutné nanovo vytvoriť aj JavaScript, ktorý obsluhoval zmenu nastavení widgetu v reálnom čase. Tieto zmeny bolo potrebné navrhnúť v časti Presenter a Viewer. Podrobnejšie sa tieto záležitosti spomínajú v časti Implementácia.

Generovanie kódu

Cieľom je aby sa vygeneroval kód widgetu, ktorý po vložení na stránku widget zobrazí s požadovanými informáciami. Táto funkcionalita bude naprogramovaná resp. preprogramovaná do frameworku Nette.
Dáta vybrané používateľom sa v presenteri uložia do premenných, tie sa potom pošlú modelu, kde sa vyskladá samotný kód widgetu. Z iframe sa zavola stará funkcionalita pre zobrazenie widgetu s jeho štýlom a informáciami, ktoré sú vytiahnuté na základe filtráciou navolených premenných, uložených v GET metóde.

REST API

Na základe požiadavky zadávateľa projektu, bola navrhnutá jednoduchá aplikácia, ktorá má umožňovať zadať vlastný text a z tohto textu volaním externej služby poskytovanej AlchemyAPI získať z daného textu kategóriu. V princípe, ktorý používa Nette framework by malo ísť o nasledujúci spôsob získania kategórie z textu. Text napísaný používateľom do textového rámca sa pomocou príslušnej metódy prezentera pošlú modelu, ktorý vykoná úpravy nad týmto textom potrebné pre odoslanie korektnej požiadavky na externý zdroj. Po obdržaní odpovede zo zdroja sa táto odpoveď vyhodnotí a ak nastane úspešná kategorizácia

textu, z modelu sa pošle do prezentera zistená kategória a prezenter ju odošle pohľadu na zobrazenie používateľovi.

Implementácia

Štýl widgetu

V predošlej časti projektu bolo možné JS funkcie aplikovať priamo na formulár pomocou príkazov onKeyUp, onChange a podobne, ktoré však nebolo možné použiť aj vo frameworku Nette. Na základe toho sa stal predošlý obsluhujúci JS nepoužiteľný a teda bolo nevyhnutné nanovo vytvoriť aj formulár aj nový obsluhujúci JS, ktorý by obsluhoval priamo objekty na stránke a nebol volaný pomocou parametrov ako tomu bolo v predošlom prípade. Tento prístup sa podarilo aplikovať a widget je teda možné používať a nastavovať mu jednotlivé parametre ako v predošlej verzii bez frameworku, ba dokonca vďaka frameworku je ošetrenie nesprávnych vstupov priamo vo formulári a teda nie je potrebné, aby tieto záležitosti riešil a ošetroval až samotný JS. Ten tak môže pracovať už iba s validnými hodnotami, ktoré priamo nastaví jednotlivým prvkom widgetu.

Aktualizovaný JS kód, ktorý umožňuje zmenu obsahu widgetu a jeho rozmerov, či textu v ňom obsiahnutom:

```
<script type="text/javascript">
//event handler for left mouse button clicked
document.onclick = ClickCheck;
function ClickCheck(e)
{
  var radioObj = document.forms['form1'].elements['template'];
  var radioLength = radioObj.length;
  for(var i = 0; i < radioLength; i++)
  {
    if(radioObj[i].checked)
    {
      //set TEMPLATE of widget
      document.getElementById("widget").style.backgroundImage              =
"url('../../"+radioObj[i].value+"')";
       document.getElementById("frmcreateStyleForm-aaaa").value = '../../'
+ radioObj[i].value;
    }
  }
}

//event handler for Keyboard pressed
document.onkeyup = KeyCheck;
function KeyCheck(e)
{
  //set TITLE of widget
  document.getElementById("name0").textContent                              =
document.getElementById("frmcreateStyleForm-nazov").value;
```

```
  //set WIDTH of widget window to value between 150 - 800
  var sirka = document.getElementById("frmcreateStyleForm-sirka").value;
  var max_sirka = 800;
  var min_sirka = 150;
  if(!isNaN(sirka))
  {
    document.getElementById("widget").style.width                          =
document.getElementById("frmcreateStyleForm-sirka").value +'px';
    if(sirka<min_sirka)
    {
      document.getElementById("widget").style.width = min_sirka+'px';
    }
    if(sirka>max_sirka)
    {
    document.getElementById("widget").style.width = max_sirka+'px';
    }
  }
  else
  {
  document.getElementById("widget").style.width = min_sirka+'px';
  }
  //end of widget WIDTH setting

  //set HEIGHT of widget window to value between 150 - 800
  var vyska = document.getElementById("frmcreateStyleForm-vyska").value;
  var max_vyska = 800;
  var min_vyska = 150;
  if(!isNaN(vyska))
  {
    document.getElementById("widget").style.height                         =
document.getElementById("frmcreateStyleForm-vyska").value +'px';
    if(vyska<min_vyska)
    {
      document.getElementById("widget").style.height = min_vyska+'px';
    }
    if(vyska>max_vyska)
    {
    document.getElementById("widget").style.height = max_vyska+'px';
    }
  }
  else
  {
  document.getElementById("widget").style.height = min_vyska+'px';
  }
  //end of widget HEIGHT setting
}
</script>
```

Generovanie widgetu

V presenteri WidgetPresenter.php si postupne uložíme vyfiltrované dáta v jednotlivých formulároch: showFilters a createStyle do session. Tie sa odošlú modelu WidgetModel.ph, ktorý z nich vyskladá kód widgetu.

Funkcia z modelu WidgetModel.php, ktorá skladá kód:

```
function writeCode($rss, $widgetName, $height, $width, $numberOfArticles,
$template, $category, $rssCategory)    //link code that represents widget
            {
                  $session = NEnvironment::getSession();
                  $namespace = $session->getNamespace('widgetData');

                  if(count($rssCategory)==0)
                  {
                        $cat = $category;
                  }
                  else
                  {
                        $cat="";
                        for($i=0;$i<count($rssCategory);$i++)
                        {
                              $cat .= "rssCat[]=".$rssCategory[$i]."&";
                        }
                  }
                  $widgetCode ="<iframe width=\"$width\" height=\"$height\"
frameborder=\"0\"              scrolling=\"no\"              name=\"widget\"
src=\"http://147.175.159.182/team10issi/beta3/trunk/widget/widget.php?";
                  if(strpos($rss,"rss.sme.sk") > 0)
                  {
                        $widgetCode
.="nazov=$widgetName&sirka=$width&vyska=$height&pocet_clankov=$numberOfArti
cles&template=$template&subject=$cat&category=";
                  }
                  else if (count($namespace->place)>0)
                  {
                        $placeString="";
                        for($i=0;$i<count($namespace->place);$i++)
                        {
                              $placeString     .=     "place[]=".$namespace-
>place[$i]."&";
                        }
                        if($cat!="")
                              $widgetCode
.="nazov=$widgetName&sirka=$width&vyska=$height&pocet_clankov=$numberOfArti
cles&template=$template&".$placeString."".$cat."";
                        else
                              $widgetCode
.="nazov=$widgetName&sirka=$width&vyska=$height&pocet_clankov=$numberOfArti
cles&template=$template&$placeString";
                  }
                  else
                  {
                        $widgetCode
.="nazov=$widgetName&sirka=$width&vyska=$height&pocet_clankov=$numberOfArti
cles&template=$template&$cat";
                  }
                  if(!empty($rss))
                  {
                        $widgetCode .= "source=RSS&RSSlink=$rss\"";
                  } else
                  {
                        $widgetCode .= "&source=Google\"";
                  }
                  $widgetCode .= "</iframe><div id=\"result\"></div>";
                  return $widgetCode;
            }
```

Vygenerovaný kód sa nakoniec pošle späť presenteru a ten ho pošle šablone na zobrazenie.

Alchemy (REST)API

Implementácia bola oproti návrhu vykonaná s miernymi zmenami a to z dôvodu doterajšej veľmi slabej praktickej skúsenosti s Nette frameworkom. Táto úloha bola teda implementovaná pomocou súboru index.html, v ktorom si používateľ zvolil text a po odoslaní textu pomocou tlačidla sa odoslala požiadavka na AlchemyAPI, pričom v odpovedi bola kategória a dodatočné informácie. Výsledkom tohto procesu bol výpis kategórie na základe zadaného textu a takisto aj informácia o presnosti, s akou táto externá služba dokázala správne určiť kategóriu textu, teda akýsi odhad správnosti, ktorý bol nadobúdal hodnoty 0,4001 = unknown, teda kategória sa nedala na základe vloženého textu určiť, alebo bola táto hodnota väčšia ako spomínané číslo a čím viac sa blížila k číslu 1, tým presnejšie bola kategória určená. Vzhľadom na jazykové obmedzenia tejto služby bol zadávaný text v anglickom jazyku.

Volanie tejto služby bolo implementované pomocou php funkcie cURL, ktorá dokáže vykonať požiadavku na základe URL adresy a vráti odpoveď. URL adresa bola vyskladaná na základe špecifikácie AlchemyAPI v podobe:
http://access.alchemyapi.com/calls/text/TextGetCategory?apikey=$key&text=$txt&outputMode=$outputMode,
kde http://access.alchemyapi.com/calls/text/TextGetCategory je volanie REST API a za otáznikom nasledujú parametre požiadavky na API, pričom ***apikey*** predstavuje jedinečný kľúč, ktorý sme získali registráciou na danej stránke, ***text*** je zadaný text, z ktorého chceme získať kategóriu a ***outputMode*** predstavuje formu, v akej je vrátený výsledok, teda odpoveď(v našom prípade bola implementovaná forma XML i JSON). V nasledujúcej ukážke kódu je vidno i získanie dodatočných informácií odpovede, kde sa využíva kontrola statusu, teda ak sa odpoveď vráti so statusom ***http_code==200***, teda OK, pracuje sa so získanými údajmi, inak program ohlási chybu. Pole ***key***(40-miestne) je vyplnené hviezdičkami v dokumentácii kvôli ochrane nadobudnutého kľúča, ku ktorého ochrane sme sa zaviazali pri registrácii.

```
function getAlchemyResponseData($data)
{
    $outputMode = "json";
    $txt =  $data;
    $txt = str_replace(" ", "%20", $txt);
    $key = "****************************************";
```

```
    $url                                                                    =
"http://access.alchemyapi.com/calls/text/TextGetCategory?apikey=$key&text=$
txt&outputMode=$outputMode";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_POST, 1);

    $responseData = curl_exec($ch);
    $info = curl_getinfo($ch);

    curl_close($ch);

    if($info['http_code']==200)
        return $responseData;
    else
        return "Error";
}


function getCategoryFromJSON($data)
{
    $category = "";
    $confidence = "";

    if($data != "Error")
    {
        $json = json_decode($data, true);
        $category = $json["category"];
        $confidence = $json["score"];
        echo   "<div   class=\"text\">Category:   $category   .   Confidence:
$confidence .</div><br/>";
    }
    else
    {
        echo       "<div       class=\"text\">Error:       request       not
processed!</div><br/>";
    }
}
```

Testovanie

V tomto šprinte sa testovali hlavne časti týkajúce sa kategorizácie textu pomocou REST služby poskytovanej pomocou Alchemy API a spracovanie údajov v samotnom widgete tak, aby obsahoval požadované údaje, hlavne hyperlinku odkazujúcu sa na pôvodný zdroj.

Tabuľka č.1: Akceptačný test pre kategorizáciu textu.

ID	1	**Názov**	Kategorizácia textu pomocou Alchemy API		
Úroveň splnenia testu			Musí – ~~Mal by~~ – ~~Mohol by~~	**Autor**	Jozef Macho
Rozhranie		Jednoduché rozhranie pre kategorizáciu			
Účel		Overenie správnej funkčnosti kategorizácie textu			
Vstupné podmienky		Text v anglickom jazyku, aspoň 5 významových slov			
Výstupné podmienky		Zobrazená kategória			
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**
1	Vloženie textu do textového okna a stlačenie tlačidla na odoslanie		Zobrazenie kategórie a pravdepodobnosti správnej kategorizácie.		Skutočná ako očakávaná.

Tabuľka č.2: Akceptačný test pre odkazovanie sa pomocou hyperlinky z widgetu na zdrojovú stránku.

ID	2	**Názov**	Odkazovanie sa pomocou hyperlinky z widgetu na zdrojovú stránku		
Úroveň splnenia testu			Musí – ~~Mal by~~ – ~~Mohol by~~	**Autor**	Jozef Macho
Rozhranie		Stránka používateľa s naším widgetom			
Účel		Overenie správnej funkčnosti odkazovania hyperlinky vo widgete			
Vstupné podmienky		Údaje vo widgete – aspoň jeden záznam			
Výstupné podmienky		Zobrazená kategória			
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**
1	Stlačenie odkazu v podobe hyperlinky vo widgete		Zobrazenie zdrojovej stránky podľa hyperlinky.		Stránka sa nezobrazí, .

Tabuľka č.3: Akceptačný test pre zobrazenie údajov pri kustomizácii wdigetu.

ID	3	**Názov**	Zobrazenie údajov pri kustomizácii widgetu		
Úroveň splnenia testu			Musí – ~~Mal by~~ – ~~Mohol by~~	**Autor**	Jozef Macho
Rozhranie		Stránka kustomizácie widgetu			
Účel		Overenie správnej funkčnosti zobrazenia údajov v náhľade widgetu pri jeho kustomizácii			
Vstupné podmienky		Správna cesta k dátovému zdroju, aspon jeden záznam v dátovom zdroji			
Výstupné podmienky		Zobrazené údaje v náhľade widgetu			
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**
1	Výber dátového zdroja, zmena parametrov widgetu vo widgetizéri		Zobrazenie widgetu podľa navolených parametrov s údajmi z dátového zdroja(titul správy, linka).		Zobrazia sa iba tituly jednotlivých správ.