Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: SOFTVÉROVÉ INŽINIERSTVO

Implementácia obrazoviek

Vývoj a implementácia grafického návrhu stránok v Ruby On Rails

Metódy softvérového inžinierstva - metodika

November 2008 Bc. Martin Valašík

1. Úvod

Účelom tohto dokumentu je definícia postupu vývoja a tvorby dizajnu webového sídla napísaného v jazyku Ruby a framework RubyOnRails. Opísaný je celý postup vývoja dizajnu dôraz je kladený na jeho implementáciu a využitie vlastností tohto frameworku.

2. Pojmy

- *Framework* – „rámec“, súbor funkcií a tried podporujúci rýchly vývoj aplikácie
- *MVC* – Model-View-Controller, architektúra, ktorá oddeľuje prezentačnú, dátovú a výpočtovú logiku.
- *Controller* – časť MVC architektúry slúžiaca na tvorbu výpočtov a ovládacích prvkov.
- *Model* – časť MVC architektúry slúžiaca na uchovávanie údajov a dátovej špecifikácie
- *View* – časť MVC architektúry slúžiaca ako prezentačná vrstva
- *ROR* – Ruby On Rails, framework napísaný v objektovom jazyku Ruby a využívjúci vzor *Model – View – Controller*.
- *Tag* – značka. V HTML sa pomocou tagov označujú jednotlivé úseky obsahu stránky.
- *HTML – Hypertext* Markup Language, značkovací jazyk určený na tvorbu hypertextových dokumentov.
- *CSS* – Cascading Style Sheets, jazyk opisujúci prezentáciu dokumentov napísaných v značkovacích jazykoch.
- *Layout* – rozloženie hlavných grafických línií stránky

3. Použitá literatúra

[1] RubyOnRails, API špecifikácia : http://api.rubyonrails.org/

4. Implementácia grafického dizajnu stránky

Vstup	Požiadavky zákazníka
Výstup	Implementovaný grafický návrh stránky
1.	Vytvorenie grafického návrhu stránky
2.	Rozloženie grafického návrhu na elementy
3.	Implementácia grafického návrhu stránky

Tab. 1 - proces implementácie grafického dizajnu stránky

Proces implementácie grafického dizajnu stránky je v tabuľke (tab.1) rozdelený na tri pod procesy, ktoré budú v najbližších podkapitolách popísané.

4.1. Vytvorenie grafického návrhu stránky

Vstup	Požiadavka zákazníka
Výstup	Grafický návrh stránky
1.	Zadefinovanie požiadaviek na dizajn stránky
2.	Hrubý návrh dizajnu
3.	Schválenie návrhu zákazníkom
4.	Dolaďovanie dizajnu stránky
5.	Finálne schválenie grafického návrhu stránky zákazníkom

Tab. 2 - proces tvorby grafického návrhu stránky

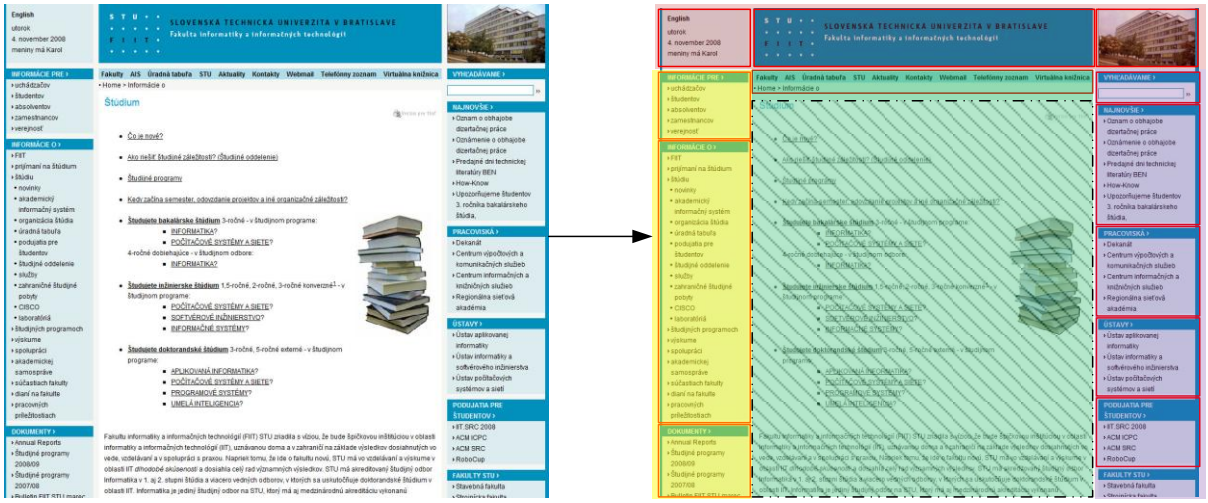
V tomto procese sa vyskytujú dva rozhodovacie kroky. Krok č.3 – Schválenie návrhu zákazníkom a krok č.5 – Finálne schválenie grafického návrhu stránky zákazníkom. V prípade kroku č.3 je potrebné sa pri neschválení vrátiť späť na krok č.2. V prípade neschválenia v kroku č.5 sa vraciame späť na krok č.4.

4.2. Analýza grafického návrhu

Vstup	Grafické návrhy stránky
Výstup	Analyzované a identifikované elementy a obsahová časť
1.	Identifikácia hlavných línií v grafickom návrhu stránky
2.	Identifikácia obsahovej časti návrhu stránky
3.	Identifikácia jednotlivých elementov, ako objektov ktoré sú logicky súdržné

Tab. 3 - proces analýzy grafického návrhu obrazovky

Proces analýzy schváleného grafického návrhu určuje objekty stránky a ich rozloženie. Na základe takejto analýzy je jasné ktoré časti grafického návrhu budú implementované samostatne a ktoré súčasne. Obrázok (obr.1) znázorňuje príklad takejto analýzy, jej vstup a výstup.



Obr. 1 – príklad analýzy grafického návrhu stránky

	Hlavné línie grafického návrhu
	Elementy v grafickom návrhu stránky
	Obsahová časť stránky

Tab. 4 - legenda k Obr. 1

4.3. Implementácia grafického návrhu stránky

Vstup	Analyzované a identifikované elementy a obsahová časť
Výstup	Zdrojové kódy dizajnu stránky

1.	Implementovanie layout-u na základe identifikovaných hlavných línií grafického návrhu
2.	Implementovanie jednotlivých identifikovaných elementov
3.	Rozdelenie zdrojových kódov dizajnu stránky do viacerých súborov
4.	Nahradenie častí kódu ROR funkciami

Tab. 5 - proces implementácie grafického návrhu stránky

Pri implementácií layout-u (krok č.1) sa pre každú líniu vytvorí HTML tag *DIV*. Každý z týchto tagov má ako parameter *id* nastavený na jedinečný identifikátor. Ten vychádza z pozície tejto línie. Tento príklad zobrazuje priradenie názvov vyplývajúci z analýzy na obrázku obr.1.

```
<HTML>
...
<DIV id="top"> ... </DIV>
<DIV id="left"> ... </DIV>
<DIV id="center"> ... </DIV>
<DIV id="right"> ... </DIV>
...
</HTML>
```

Takto identifikované časti majú následne v CSS popísanú ich pozíciu a vizualizáciu.

```
#top { … }
#left { float: left; …}
#center { float: left; … }
#right { … }
```

Podobne ako pri implementácii layout-u tak aj pri implementácii jednotlivých elementov sa pre každý element vytvorí tag *DIV*. V tomto prípade parameter *id* vychádza z kontextu elementu. Pri samotných elementoch je potrebné pridať aj hodnotu parametru *class*. Ten vychádza z faktu, že mnohé elementy majú rovnakú alebo podobnú vizuálnu podobu. Tým, ktoré ju majú podobnú, sa priradí rovnaká CSS trieda. Takto je možné pomocou CSS kontrolovať vizuálnu podobu elementov. Pre špecifikovanie vizuálnej stránky tagov v elementoch alebo triedach sa špecifikujú v CSS jednotlivé elementy na úrovni daného elementu alebo triedy.

```
...
<DIV id="informations_for" class="side_element"> ... </DIV>
<DIV id="informations_about" class="side_element"> ... </DIV>
...
```

```
.side_element { … } /* trieda vizualizacie */
#informations_for {  … } /* špecifikovanie vizualizacie elementu */
#informations_about { … }
.side_element ul { … } /* vizualizácia tagu ul v rámci triedy vizualizácie
```

4.4. Rozdelenie zdrojových kódov dizajnu stránky do viacerých súborov

RubyOnRails je založený na MVC architektúre. Jednotlivé *Views* sú previazané na konkrétne *controller-y* a teda zabezpečujú obsahovú časť stránky. Nasledujúci blok znázorňuje štruktúru vizualizácií v RubyOnRails. Všetky implementované *layout-y* sa musia nachádzať v priečinku /views/layouts/.

ROOT :
|-> Views
    |-> layouts –priečinok obsahujúci jednotlivé identifikované *layout-y*
        |-> *layout1.html.erb* – konkrétny *layout*
        |-> …
    |-> *controller* – priečinok obsahujúci *views* súvisiace s konkrétnym *controller-om*
        |-> action1.html.erb – *view* pre obsahovú časť vytvorenú akciou *action1*
        |-> _form.html.erb - zobrazovacia časť súvisiaca s konkrétnym *controller-om*
        |-> …
    |-> shared
    |-> …
|

Obsahové časti stránok sú vytvárané volaniami akcií (*actions*) niektorého *controller-u*. Každá z týchto akcií má priradený aj konkrétny *view*. Tento je umiestnený v priečinku *views* daného *controller-u*. Názov tohto *view* zodpovedá názvu akcie, ktorá bola nad daným *cintrollerom* zavolaná.

Podmienka	Umiestnenie
layout stránky	/views/layouts/
obsahová časť stránky	/views/:controller[1]/

Tab. 6 - umiestnenie objektov do adresárovej štruktúry ROR

Pre potreby implementácie elementov a ich volania v rôznych častiach je potrebné vytvoriť priečinok *Shared*. Pre jednotlivé typy elementov je určená ich pozícia v adresárovej štruktúre na základe ich použitia. Jednotlivé pravidlá sú zoradené podľa priority a v prípade konfliktu pravidiel sa vyberá pravidlo s vyššou prioritou.

[1] „:controller“ určuje priečinok s *views* pre konkrétny *controller*

Názov elementu je *„_nazov_elementu.html.erb“*, pričom názov elementu vychádza z kontextu elementu a je totožný s *id* atribútom jeho tagu *DIV*.

#	Podmienka	Umiestnenie
1.	element priamo a obsahovo súvisí s prácou *controller-u*	/views/:controller/
2.	Element, na ktorý sa odkazuje z *layout-u*	/views/shared/
3.	element, na ktorý sa odkazuje z viacerých *views* rôznych *controller-ov*	/views/shared/
4.	element, na ktorý sa odkazuje iba *views* z jedného *controlleru*	/views/:controller/

Tab. 7 - pravidlá pre umiestňovanie elementov do adresárovej štruktúry ROR

Vnáranie obsahu jednotlivých vizuálnych objektov sa vykonáva prostredníctvom funkcií v RubyOnRails. Konkrétne funkcie a prípady ich použitia uvádza tabuľka č.8.

Prípad	Funkcia
volanie obsahu stránky	yield
volanie elementu umiestneného v tom istom priečinku	renrer :partial “nazov_elementu“
volanie elementu umiestneného v inom priečinku	renrer :partial “priecinok/nazov_elementu“

Tab. 8 - funkcie pre vnáranie obsahu jednotlivých objektov

Názov elementu v tomto prípade zodpovedá jeho názvu bez prefixu „_“ a koncovky „.html.erb“.

Nahradzovanie častí kódu ROR funkciamibude popísané v zvláštnej kapitole.

5. Nahradzovanie častí kódu RubyOnRails funkciami

Framework RubyOnRails ponúka vývojárom, okrem iného, aj možnosť písať časti HTML kódu pomocou vlastných, prívetivejších funkcií. Tie sú vytvorené pre účely generovania kódu, ktorý pracuje priamo s aplikáciou.

Presnú špecifikáciu všetkých ROR funkcií, ktoré budú spomenuté, je možné nájsť na http:// api.rubyonrails.org/.

5.1. Navigácia

V tejto časti sú opísané ROR funkcie, ktoré súvisia s odkazovaním sa na dokumenty alebo objekty.

HTML	ROR
A : (rôzne variácie použitia)	
obyčajný odkaz	liink_to
vytvorenie odkazu ak platí podmienka	link_to_if
vytvorenie odkazu ak neplatí podmienka	link_unless
vytvorenie odkazu ak sa nezhoduje aktuálna URI s vygenerovanou URI	link_unless_current
odkaz vytvárajúci AJAX volanie	link_to_remote
odkaz na JavaScript funkciu	link_to_function
odkaz na e-mail	mail_to

Tab. 9 - nahradzovanie HTML tagov ROR funkciami – navigácia

Správna ROR funkcia sa vyberá na základe konkrétneho použitia.

5.2. Formuláre

Táto časť opisuje spôsob nahradzovania HTML tagov týkajúcich sa formulárov. Tie vďaka týmto funkciám budú správne spolupracovať s aplikáciou napísanou v ROR. Funkcie v tomto prípade rozdeľujeme do dvoch skupín :

1. Pracujúce v kontexte s objektom namodelovaným v ROR
2. Získavajúce dodatočné informácie, a teda nesúvisia s nejakým objektom v ROR

V prípade možnosti 2. je ale potrebné zadať názov premennej vlastnoručne. Tabuľka č.9 zobrazuje spôsob nahradzovania HTML tagov formulárov. V ľavom stĺpci sú HTML tagy. Tým sa určí, či sú v kontexte s niektorým objektom alebo nie a následne sú nahradené potrebnou ROR funkciou.

HTML	ROR s modelom	ROR bez modelu
input tagy : (rôzne typy definované podľa argumentu *type*)		
checkbox	check_box	check_box_tag
file	file_field	file_field_tag

hidden	hidden_field	hidden_field_tag
password	password_field	password_field_tag
radio	radio_button	radio_button_tag
text	text_field	text_field_tag
Iné tagy formulára :		
form	form_for	form_tag
textarea	text_area	text_area_tag
select	select	select_tag

Tab. 10 - nahradzovanie HTML tagov ROR funkciami - formuláre

Nasledujúci blok zobrazuje použitie ROR funkcie v prípade, že je v kontexte s objektom namodelovaným v ROR a aj v prípade, že so žiadnym takýmto objektom nie je v kontexte.

```
Prípad kedy formulár súvisí s objektom :
<% form_for :person, @person, :url => { :action => "update" } do |f| %>
   First name: <%= f.text_field :first_name %>
   Last name : <%= f.text_field :last_name %>
  <% end %>

Prípad kedy objekt nesúvisí s objektom :
...
   Confirm    : <%= check_box_tag "other[confirm]"  %>
...
```

5.3. Iné opakujúce sa HTML tagy

Táto časť opisuje prípad, kedy je potrebné nahradiť iné HTML tagy ROR alternatívami. Tento prípad nastáva v momente, kedy je potrebné zobraziť zoznam objektov tak, že každý objekt je vnorený do práve jedného tagu, ktorý majú všetky objekty rovnaký.

HTML	ROR s modelom
Rôzne tagy : (názov tagu je *nazov_tagu*)	
tag	content_tag_for :*nazov_tagu*

Tab. 11 - nahradzovanie HTML tagov ROR funkciami - iné, opakujúce sa

Prácu s touto funkciou popisuje nasledujúci blok.

ROR funkcia :

content_tag_for(:li, @person, :class => "bar") ...

Výsledok :

*<li id="person_1" class="person bar"> ... *

*<li id="person_2" class="person bar"> ... *

...