Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Askalot meets Harvard Courses at edX

[Askalot2edX]

Metodika pre vývoj

Tím: číslo 6, AskEd
Vedúci tímu: Ing. Ivan Srba
Členovia tímu: Černák Martin, Gallay Ladislav, Hnilicová Eva, Huňa Adrián, Jandura Filip, Žuffa Tibor
Akademický rok: 2015/2016
Autor: Adrián Huňa
Verzia číslo: 3
Dátum poslednej zmeny: 14.10.2015

1. Úvod

Cieľom tohto dokumentu je zadefinovanie základných postupov pre jednotnú tvorbu a komentovanie zdrojových kódov. Metodika zahŕňa konvencie písania zdrojového kódu, ktoré sú jeho tvorcovania povinní dodržiavať. Uvedená metodika je zameraná na programovací jazyk ruby a rámec (angl. framework) Ruby on Rails[1].

Nakoľko preberáme kód po predchádzajúcom tíme, uvedené metodiky sú vo väčšine prípadov zhodné s metodikami, ktoré používal tím naRuby[2] v akademickom roku 2013/2014 pri vývoji systému Askalot.

1.1. Zadefinovanie pojmov

Tab. 1 *Použité pojmy s vysvetlením.*

Pojem	**Vysvetlenie**
meno	Meno autora (značky, kódu).
správa	Vyjadruje bližší opis značky (odôvodnenie).
snake_case	Kovencia alebo tiež štýl písania viacslovných názvov, kde každé slovo začína malým písmenom a slová sú navzájom oddelené znakom _ (podtržníkom).
CamelCase	Konvencia alebo tiež štýl písania viacslovných názvov, kde každé slovo začína veľkým písmenom a jednotlivé slová nie sú navzájom oddelené.
SCREAMING_SNAKE_CASE	Konvencia alebo tiež štýl písania viacslovných názvov, kde každé písmeno je písané veľkým písmenom a jednotlivé slová sú navzájom oddelené znakom _ (podtržníkom).

[1] http://rubyonrails.org/
[2] http://labss2.fiit.stuba.sk/TeamProject/2013/team13is-si/

2. Postupy

Používaným jazykom pri písaní kódu ako aj komentárov je výhradne angličtina. Žiadne iné jazyky nie sú povolené.

Komentovanie a značkovanie zdrojového kódu

Komentáre

- V zdrojovom kóde nie sú povolené žiadne komentáre okrem dokumentačných komentárov. Z tohto dôvodu je potrebné písať kód tak, aby bol jasný na prvý pohľad.
- Každá zakomentovaná časť zdrojového kódu musí mať pri sebe značku (viď nižšie) s odôvodnením.
- Za značkou komentára nechávať jednu medzeru.
    - Príklad:
      ```
      #zle
      # dobre
      ```

Značky

- Značka sa do kódu píše rovnako ako komentár v tomto tvare: {značka} ({meno}) {správa}.
- Značka má byť stručná, ale jasná.
- V správe k značke sa nahradzuje:
    - zmazať kód (angl. remove code) skratkou rm,
    - presunúť kód (angl. move code) skratkou mv.
- Povolené značky v kóde sú uvedené v Tab. 2.

Tab. 2 *Použité pojmy s vysvetlením.*

Značka	**Význam**
TODO	Potrebné vyriešiť
FIX	Potrebné opraviť

Príklad:

```
# TODO (huna) rm
…
<!-- TODO (huna) mv
…
```

Zdrojový kód v jazyku HTML

- Vždy musia byť použité `"` namiesto `'`.
- Odsadzuje sa dvoma medzerami.
- Pravidlá pre rámec Twitter Bootstrap[3]:
    - Element triedy `row` môže obsahovať len elementy triedy `col`.

[3] getbootstrap.com

Zdrojový kód v jazyku JavaScript

- Používa sa jazyk CoffeeScript[4].
- Odsadzuje sa dvoma medzerami.

Zdrojový kód v jazyku ruby

Formátovanie zdrojového kódu

- Súbory musia byť kódované v UTF-8.
- Na konci riadkov vždy vložiť \n znak.
- Na začiatku riadka použiť odsadenie 2 medzery.
- Na odsadenie použiť 2 medzery. Tabulátor sa nesmie používať.
- Nikdy nepoužívať bodkočiarku.
- Vkladať medzeru okolo operandov a znaku rovná sa.
  - Príklad: x = 1 + 2
- Nepoužívať vnútorné medzery pri `()` a `[]`.
- Nepoužívať vnútorné medzery okolo výrazov v reťazcoch `"hello #{expression}"`.
- Vnútorné medzery používať okolo hash literálov alebo blokov `{ a: 1 }`, s výnimkou v prípade ako `{ … { … } }`, ktorý sa upravuje na `{ … { … }}`.
- Vložiť voľný riadok medzi definíciami `def`, `class`, `module` a pod..
- Nedávať biely znak medzi funkciu a zoznam argumentov.
- Nenechať biele znaky na konci riadkov.
- Nenechávať dva a viac voľných riadkov po sebe.
- `When` vnoriť tak hlboko ako `case`, nie o úroveň hlbšie.
- Nepoužívať priame priradenie z jazykovej konštrukcie, t.j. nepoužívať `result = if … .`
- Po priradení by mal ísť prázdny riadok.
- Pred `return` by mal ísť prázdny riadok.
- Používať voľné riadky okolo `next`.

Syntax zdrojového kódu

- Prísny zákaz používať for, then, and, or, not.
- Pri definícii metód používať ( ), iba ak má metóda nejaké argumenty.
- Zátvorky ( ) použiť iba ak sprehľadnia kód alebo si to vyžaduje syntax jazyka.
- Preferovať ternárny operátor oproti `if/then/else/end`
  - Príklad:

```
# zle
if a < b
  then
    a
  else
    b

# dobre
a < b ? a : b
```

- Vhodne používať if/unless.
  - Príklad:

```
# zle
if !vyraz
# dobre
```

[4] http://coffeescript.org/

```
unless vyraz
```

- Nepoužívať `unless` spolu s `else`, pozitívny prípad bude vždy ako prvý.
- Používať radšej `loop` s `break` ako `begin/end/until`, resp. `begin/end/while`.
- Vždy používať {} pre:
  - jednoriadkové bloky,
  - bloky, na ktoré nadväzuje volanie.
    - Príklad:

```
{ "jednoriadkovy blok" }
# blok, na ktory nadvazuje volanie
{ … }.join
```

- Nepoužívať `return` a `self` pokiaľ to nie je nutné.
- Nepoužívať operátory === a `->`.
- Nepoužívať $ (globálne) premenné.
- Používať `proc` namiesto `Proc.new`.
- Používať _ pre nepotrebné premenné
  - Príklad:

```
x = hash.map { |_, v| v + 1 }
```

- Používať Array.join a nie *.
- Použiť radšej `(1000..2000).include?(x)` alebo `x.between?(1000..2000)` namiesto `x >= 1000 && x <= 2000`.
- Používať predikáty namiesto ==
  - Príklad:

```
# zle
x == nil
# dobre
x.nil?
```

- Vyhýbať sa používaniu vnorených podmienok pri riadení toku programu.
- Nepoužívať

```
if cond return x
else return y
```

  ale

```
return x if cond
return y
```

- Nepoužívať

```
return nil if cond
```

  ale

```
return x unless cond
```

Pomenovanie v zdrojovom kóde

- Pomenovať veci presne, ideálne jednoslovne.
- Nepoužívať skratky dlhšie ako jedno písmeno (jednopísmenové skratky ale používať opatrne, odporúča sa použiť ich iba v blokoch). Napríklad žiadne `fld`, ale celým slovom `field` alebo skratka `f`.
- Povolené viacpísmenové skratky sú:
  - `args`
  - `params`

- Používať `map` miesto `collect`, `find` namiesto `detect`, `select` namiesto `find_all`, `reduce` namiesto `inject`, `reverse_each` namiesto `reverse.each`, `users.map(&:songs)` namiesto `users.map { |user| user.songs }`.

Tab. 2 *Rozdelenie štýlov pomenovania pre jednotlivé konštrukcie jazyka.*

Štýl	**Použitie pre**
snake_case	Symboly
snake_case	Metódy
snake_case	Premenné
CamelCase	Triedy
CamelCase	Moduly
SCREAMING_SNAKE_CASE	Konštanty

- Predikáty nemajú prefix is, ale sufix ?.
    - Príklad:
```
# zle
user.is_single
#dobre
user.single?
```

Narábanie s výnimkami v zdrojovom kóde

- Používať `fail` namiesto `raise`, používať `raise` iba v prípade znovuvyhodenia zachytenej výnimky.
- Používať `fail 'sprava'` namiesto `fail RuntimeError, 'sprava'`.
- Používať `fail NejakaVynimka, 'sprava'` namiesto `fail NejakaVynimka.new('sprava')`.
- Nepoužívať `return` v `ensure` bloku.
- Používať `def/rescue/end` namiesto `def/begin/rescue/end/end`.
- Nikdy nepotláčať výnimky.
- Zákaz používať výnimky pre riadenie toku programu.

Práca s množinami v zdrojovom kóde

- Používať `[]` a `{}` na vytvorenie poľa, resp. hash poľa namiesto `Array.new` a `Hash.new`.
- Používať priamo polia namiesto %w(). %i() a pod..
- Používať `first, second, …, last` namiesto `[0], [1], .., [-1]`.
- Používať `Set` namiesto `Array`, ak je to vhodné.
- Používať ako kľúče symboly namiesto reťazcov.
- Vždy pokiaľ je to možné použiť `{ one: 1}` namiesto `{ :one => 1 }`.

Práca s reťazcami v zdrojovom kóde

- Používať `''` pre reťazce namiesto `""`.
    - Ak ide o reťazce, ktoré predstavujú jazykové preklady používať `""` .
- Pre spájanie reťazcov preferujte << namiesto +.