Metodika pre vývoj RubyOnRails

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.

Zadefinovanie pojmov

V Tab. 1 sa nachádza zoznam použitých pojmov aj s ich 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 písmom a jednotlivé slová sú navzájom oddelené znakom '_' (podtržníkom).

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.

Značka	Význam
TODO	Potrebné vyriešiť
FIX	Potrebné opraviť

Príklad:

```
# TODO (Pecher) rm
```

```
<!-- FIX (Kovacik) invalid html -->
```

Zdrojové kódy

Zdrojový kód v jazyku HTML

- Vždy musia byť použité " namiesto '.
- Odsadzuje sa dvoma medzerami.
- HTML musí byť ***validné***.
- Pravidlá pre rámec Twitter Bootstrap:
    - Element triedy row môže obsahovať len elementy triedy col.

Zdrojový kód v jazyku JavaScript

- používa sa jazyk CoffeeScript
- 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

- Preferovať *each* oproti *for*
- Preferovať && a || oproti and a or kvôli prioritám operátorov.
- 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
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 pokiaľ to nie je nutné.

- Používať self pri volaní vlastných metód.
- Nepoužívať operátor ===
- Nepoužívať $ (globálne) premenné.
- 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

```
# zle
if cond return x
else return y
# dobre
return x if cond
return y
```

- Nepoužívať

```
# zle
return nil if cond
# dobre
return x unless cond
```

Pomenovanie v zdrojovom kóde

- Pomenovať veci presne, ideálne jednoslovne.
    - Príklad:

```
# zle
trumpova_politika_prospieva_americkej_ekonomike =
false
# dobre
beneficial = false
```

- 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

Prehľad štýlov pomenovania jednotlivých konštrukcií jazyka je uvedený v Tab. 3

Značka	Význam
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?
```

- Databázove boolean stĺpce majú prefix is_, has_, can_ alebo check_
    - Príklad:

```
is_active
has_group_owner
check_invoices_sum
can_auto_assign
```

- Vlastné validačné metódy začínaju prefixom *validate_*
    - Príklad:

```
validates :validate_stuff
```

- Validácie namiesto do blokov, písať do metód

```
# zle
validates do
  ...
end

# dobre
validates :validate_stuff
```

Narábanie s výnimkami v zdrojovom kóde

- Používať fail alebo raise, pre vyhodenie výnimky
- Používať fail 'sprava' namiesto fail RuntimeError, 'sprava'
- Používať fail NejakaVynimka, 'sprava' namiesto fail NejakaVynimka.new('sprava')
- Nepoužívať return v ensure bloku.
- 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 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 "".
- Pre spájanie reťazcov preferujte:

```
# dobre
"#{var1} nieco #{var2}"
# zle
var1 + " nieco " + var2
```