

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

Kolaboratívne prototypovanie [CollabUI] Metodika písania kvalitného kódu

Tím: 24. CollabUI

Vedúci tímu: Ing. Eduard Kuric, PhD.

Členovia tímu: Adrián Nagy, Lukáš Vrba, Ján Kleň, Michal Melúch, Tomáš Mňačko, Peter Písecký, Miloslav Smetana

Študijný program: Inteligentné softvérové systémy

Ročník: 2017/2018

1. Úvod

Metodika obsiahnutá v tomto dokumente predpisuje pravidlá pre písanie kódu v jazykoch PHP a JavaScript. Cieľom tejto metodiky je zjednotiť štýly písania kódu medzi jednotlivými členmi tímu tak, aby vyprodukovaný kód bol prehľadný a nasledoval konvencie písania kódu v jazykoch PHP a JavaScript, ako aj konvencie technológií CakePHP a Node.js.

2. Dôležité odkazy

PSR-2: Coding Style Guide - <http://www.php-fig.org/psr/psr-2/>

CakePHP: Coding Standards - <https://book.cakephp.org/3.0/en/contributing/cakephp-coding-conventions.html>

CakePHP Code Sniffer - <https://github.com/cakephp/cakephp-codesniffer>

JSHint - <http://jshint.com/>

3. Pravidlá písania kódu v jazyku PHP

Kód napísaný v jazyku PHP musí nasledovať štandard [PSR-2](#), ako aj [dodatočné pravidlá, ktoré sú uvedené v dokumentácii CakePHP](#). Nižšie uvedený zoznam poskytuje prehľad najdôležitejších pravidiel pre písanie kódu v jazyku PHP.

Odsadenie

Pre odsadenie PHP kódu sa používajú 4 medzery.

Maximálna dĺžka riadku

Maximálna dĺžka riadku je 100-120 znakov.

Porovnávanie

Porovnávanie hodnôt by malo byť vždy čo najstriktnejšie. Hodnota, voči ktorej sa porovnáva by mala byť vždy umiestnená na pravej strane.

Pomenovanie metód

Názov metódy musí byť vždy napísaný vo formáte camelBack.

Volanie funkcií

Volanie funkcií by nemalo obsahovať medzeru medzi názvom funkcie a zátvorkami. Medzi každým argumentom volanej funkcie by sa mala nachádzať práve jedna medzera.

Definovanie funkcií

Parametre s predpísanou hodnotou musia byť umiestnené na konci zoznamu parametrov definovanej funkcie. Každá funkcia by *mala* niečo vracať.

Pomenovanie tried

Názvy tried musia byť napísané vo formáte CamelCase.

Pomenovanie premenných

Všetky názvy premenných musia byť vo formáte camelBack.

Definovanie konštánt

Názvy konštánt by vždy mali byť definované pomocou veľkých písmen.

Komentovanie a nápovedy pri písaní

Každá funkcia by mala byť okomentovaná tak, aby jej správanie bolo jednoducho pochopiteľné. Verejné funkcie musia obsahovať prvky umožňujúce poskytovanie nápovede pri písaní kódu v IDE.

Zreťazenie metód

Zreťazené volania metód by mali obsahovať každé volanie na samostatnom riadku odsadené 4 medzerami.

PHP značky

Vždy musí byť využití dlhší formát PHP značiek (`<?php ... ?>`).

Pomenovanie súborov

Názvy súborov neobsahujúcich triedy musia byť napísané malým písmom a predelené podtržníkom: `dlhy_nazov_suboru.php`

3.1. Kontrola napísaného kódu

Framework CakePHP poskytuje nástroj [CakePHP Code Sniffer](#) pre kontrolu správneho nasledovania vyššie uvedených štandardov.

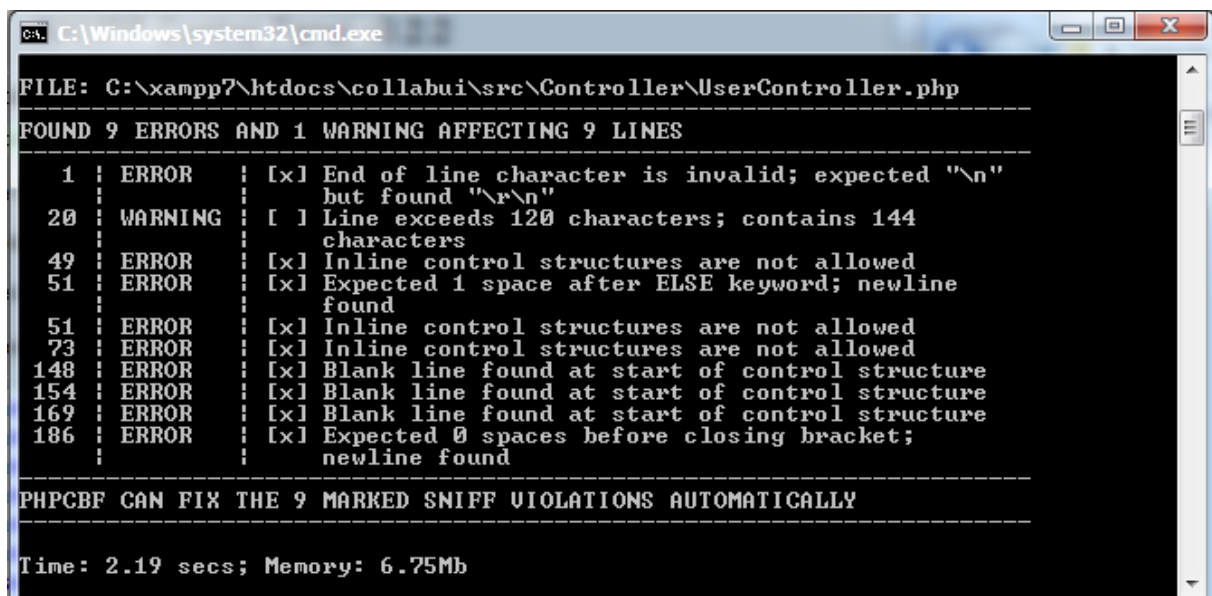
Inštalácia:

```
composer require --dev "cakephp/cakephp-codesniffer"
```

Použitie:

```
vendor/bin/phpcs --standard=CakePHP /path/to/code
```

Príklad výstupu:



```
C:\Windows\system32\cmd.exe
FILE: C:\xampp7\htdocs\collabui\src\Controller\UserController.php
-----
FOUND 9 ERRORS AND 1 WARNING AFFECTING 9 LINES
-----
 1 | ERROR   | [x] End of line character is invalid; expected "\n"
   |         | but found "\r\n"
20 | WARNING | [ ] Line exceeds 120 characters; contains 144
   |         | characters
49 | ERROR   | [x] Inline control structures are not allowed
51 | ERROR   | [x] Expected 1 space after ELSE keyword; newline
   |         | found
51 | ERROR   | [x] Inline control structures are not allowed
73 | ERROR   | [x] Inline control structures are not allowed
148| ERROR   | [x] Blank line found at start of control structure
154| ERROR   | [x] Blank line found at start of control structure
169| ERROR   | [x] Blank line found at start of control structure
186| ERROR   | [x] Expected 0 spaces before closing bracket;
   |         | newline found
-----
PHPCBF CAN FIX THE 9 MARKED SNIFF VIOLATIONS AUTOMATICALLY
-----

Time: 2.19 secs; Memory: 6.75Mb
```

Obrázok 1: Príklad výstupu nástroja CakePHP Code Sniffer

4. Pravidlá písania kódu v jazyku JavaScript

Technológia Node.js nemá predpísaný štýl písania kódu. Nižšie uvedený zoznam predpisuje nami stanovené pravidlá pre písanie kódu v jazyku JavaScript.

ECMAScript

JavaScript kód napísaný pre Node.js časť aplikácie môže využívať nové konštrukty špecifikácie ES6.

Odsadenie

Pre odsadenie JavaScript kódu sa používajú 2 medzery.

Maximálna dĺžka riadku

Maximálna dĺžka riadku je 100-120 znakov.

Porovnávanie

Porovnávanie hodnôt by malo byť vždy čo najstriktnejšie. Hodnota, voči ktorej sa porovnáva by mala byť vždy umiestnená na pravej strane.

Pomenovanie funkcií

Názov funkcie musí byť napísaný vo formáte camelBack. Výnimka: Funkcie slúžiace ako konštruktory (prototyp) musia byť napísané v CamelCase.

Volanie funkcií

Volanie funkcií by nemalo obsahovať medzeru medzi názvom funkcie a zátvorkami. Medzi každým argumentom volanej funkcie by sa mala nachádzať práve jedna medzera.

Definovanie funkcií

Parametre s predpísanou hodnotou musia byť umiestnené na konci zoznamu parametrov definovanej funkcie. Každá funkcia by *mala* niečo vracať.

Pomenovanie premenných a atribútov objektov

Všetky názvy premenných a atribútov objektov musia byť vo formáte camelBack.

Definovanie konštánt

Názvy konštánt by vždy mali byť definované pomocou veľkých písmen. Výnimka: Konštanty obsahujúce Node.js moduly volané pomocou funkcie *require*.

Komentovanie a nápovedy pri písaní

Každá funkcia by mala byť okomentovaná tak, aby jej správanie bolo jednoducho pochopiteľné. Verejné funkcie musia obsahovať prvky umožňujúce poskytovanie nápovede pri písaní kódu v IDE.

Zreťazenie metód

Zreťazené volania metód by sa mali (ale nemusia) nachádzať na samostatnom riadku odsadené 2 medzerami.

4.1. Kontrola napísaného kódu

Pre kontrolu správnosti kódu napísaného v jazyku JavaScript je možné využiť správne nakonfigurovaný nástroj [JSHint](#).

Inštalácia:

```
npm install -g jshint
```

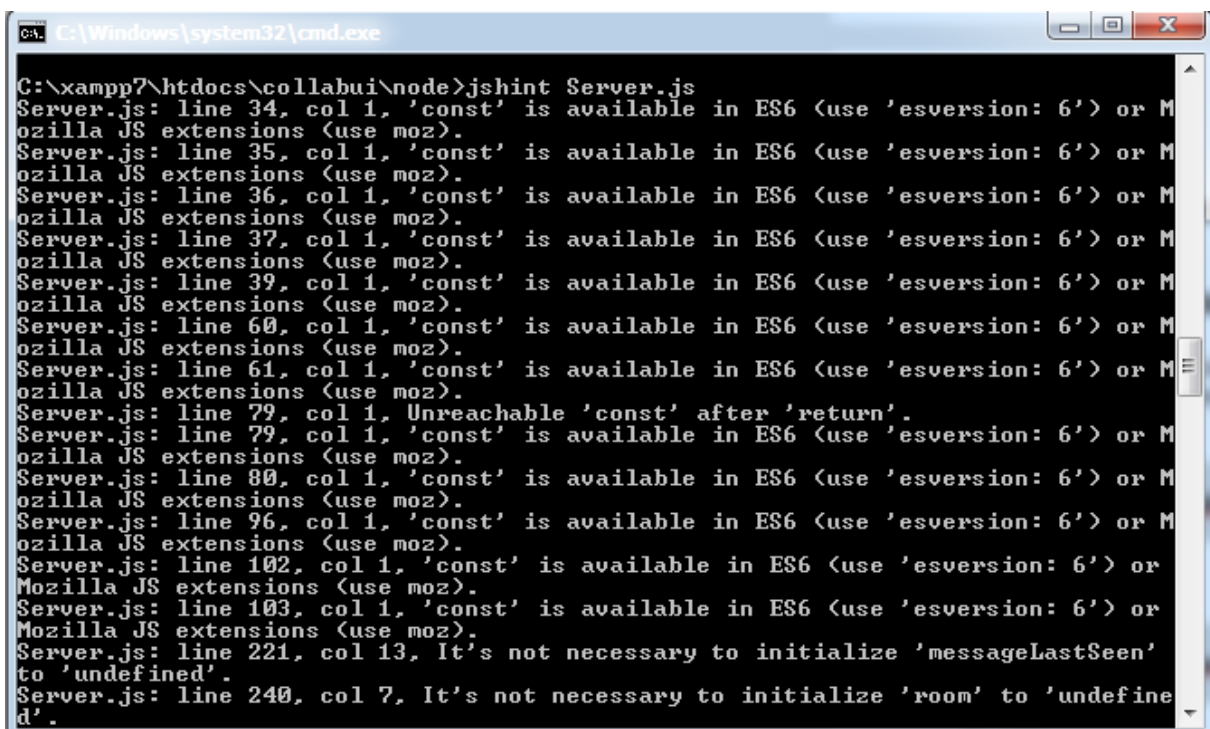
alebo:

```
npm install --save-dev jshint
```

Použitie:

```
jshint /path/to/code
```

Príklad výstupu:



```
C:\Windows\system32\cmd.exe
C:\xampp7\htdocs\collabui\node>jshint Server.js
Server.js: line 34, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 35, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 36, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 37, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 39, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 60, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 61, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 79, col 1, Unreachable 'const' after 'return'.
Server.js: line 79, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 80, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 96, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 102, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 103, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 221, col 13, It's not necessary to initialize 'messageLastSeen' to 'undefined'.
Server.js: line 240, col 7, It's not necessary to initialize 'room' to 'undefined'.
```

Obrázok 2: Príklad výstupu nedostatočne nakonfigurovaného nástroja JSHint