Trường Đại Học Bách Khoa Thành Phố Hồ Chí Minh
Chương trình Thực tập công nghiệp
Công ty Phát Triển Toàn Cầu
www.NETgroup.com.vn

99xu Coding Convention

Tác giả: Lê Quốc Nam

Thành phố Hồ Chí Minh, ngày 10, tháng 3, năm 2013

Để phù hợp với xu hướng mở rộng tới thị trường công nghệ thế giới, kể từ năm 2015 Công ty Phát Triển Toàn Cầu đã lấy tên giao dịch là NETGROUP

Mục lục

I. GIỚI THIỆU

Coding style, hay *coding convention*, là các thuật ngữ mà có lẽ bất kì một lập trình viên nào cũng từng nghe nói tới. Dù trên thực tế, việc có tuân thủ theo *coding convention* hay không hầu như không ảnh hưởng đến kết quả công việc. Hàm vẫn chạy, code vẫn thực thi đúng, và sản phẩm sẽ chẳng có gì khác biệt.

Tuy nhiên, với các hệ thống lớn, được thiết kế và bảo trì qua nhiều giai đoạn, với nguồn nhân lực khác nhau, thì việc tuân theo một *coding convention* thực sự là vấn đề sống còn. Người đọc code chắc chắn sẽ hạnh phúc và tiết kiệm thời gian hơn rất nhiều, khi đọc một đoạn code với các biến được đặt tên rõ ràng như *$userName*, *$job*, … thay vì *$a*, *$b*, *$c* (biến đặt tên theo ngôn ngữ php).

Tài liệu này sẽ qui định cho mọi người các điều phải tuân thủ khi lập trình trong hệ thống 99xu hiện tại, qua đó tạo sự thống nhất giữa các nhóm với nhau, cũng như tạo điều kiện thuận lợi cho người đi sau trong việc bảo trì, phát triển hệ thống.

II. QUI ĐỊNH VỀ CODE

1. Qui định chung về đặt tên

Mọi cái tên đều phải bằng tiếng anh, đúng chính tả, và có nghĩa.

Tên ở đây bao gồm ***tên class***, ***tên biến***, ***tên hằng số***, ***tên hàm***, ***tên file***.

- *Viết tên bằng tiếng anh* trong lúc lập trình, gần như là yêu cầu bắt buộc khi bước vào làm ở bất kì công ty nào. Hãy tập cho mình một thói quen chuyên nghiệp, từ những việc nhỏ nhất.

- *Đúng chính tả*, có thể xem đây là một cách để mọi người luyện thêm vốn từ vựng của mình.

- *Có nghĩa*, tức là tên phải gắn liền với đặc điểm, chức năng của đối tượng tương ứng. Ví dụ: *$userName* để thể hiện tên người dùng, *$image* để thể hiện ảnh đại diện…

Chú ý rằng, các *biến chạy* trong vòng lập (for, while) không cần tuân theo quy tắc có nghĩa này, vì trên thực tế, các biến *$i*, *$j* đã trở thành chuẩn.

2. Qui định về Naming convention

- Trong file php: Chúng ta sẽ sử dụng chuẩn của Java. Cụ thể như sau:

Khai báo	Qui định	Ví dụ
Class	Tên class sẽ theo dạng `UpperCamelCase`, **viết hoa chữ đầu tiên của mỗi từ.**	• `class Raster;` • `class ImageSprite;`
Hàm	Tên hàm sẽ theo chuẩn `lowerCamelCase`, **từ đầu tiên viết thường, các từ tiếp sau viết hoa**	• `run();`

	chữ cái đầu.	• `runFast();` • `getBackground();`
Tên biến	`Tương tự như tên hàm.`	• `int i;` • `int currentUserLog;` • `float myWidth;`
Hằng số	**Viết hoa tất cả kí tự, các từ cách nhau bằng dấu "_"**	• `final static int MAX_PARTICIPANTS = 10;`

- Với code javascript: Tương tự như trên.

- Với html: việc đặt tên trong code html chỉ có cho thuộc tính id và class name, chúng ta thống nhất tên sẽ viết thường, các từ cách nhau bằng dấu "_"

Ví dụ:

```
<input id="search " type="text" size="100px" name="search_box" /><br>
```

- Với tên file, chúng ta sẽ sử dụng tương tự như cách đặt tên trong html.

Ví dụ: *product_edit.php, user_login.php, cache_product.html…*

III. DOCUMENT KHI CODE

1. Giới thiệu

Nếu có ai hỏi bạn, tại sao tất cả ngôn ngữ lập trình, đều có kí tự để *"comment"* một dòng, hoặc một đoạn code (ví dụ trong Java hay C++ là // và /* */, trong Visual Basic là ' ,…). Có lẽ câu trả lời đầu tiên của không ít người là để tạm thời xóa bỏ đi đoạn code không muốn sử dụng.

Nhưng đó chỉ là một phần của vấn đề, trên thực tế, ý nghĩa quan trọng hơn rất nhiều của các kí tự này là để giải thích, ghi chú, hay nói chung là *document* cho code. Hãy thử tượng tượng bạn sẽ khó chịu thế nào, khi sử dụng một hàm nào đó của Java mà lại không có bất kì lời giải thích nào khi rê chuột vào tên hàm!

Vì thế, hãy tập cách *document* đầy đủ cho bất kì sản phẩm nào của mình khi code, vì có thể, bạn đã giúp ích rất nhiều cho người đi sau.

Sau đây là một số qui tắc mà chúng ta sẽ thống nhất khi code trong hệ thống.

2. Các qui tắc

a) Qui tắc chung

Document phải bằng tiếng anh, đúng chính tả và càng rõ ràng càng tốt.

b) Với class và hàm

Document theo chuẩn, với mọi class và hàm.

Ví dụ:

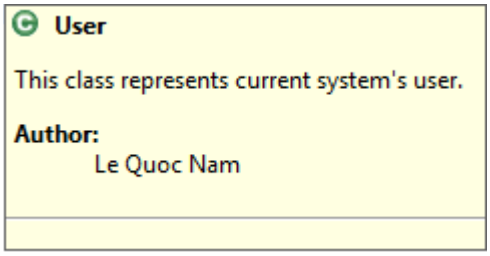
- Bạn cần viết một class thể hiện một người dùng:

```
/**
* This class represents current system's user.
* @author Le Quoc Nam
*
*/
class User{

}
```

Dòng đầu tiên chính là lời giải thích cho class, hãy giải thích càng rõ càng tốt, bằng tiếng anh, và… đúng chính tả.

Các dòng phía sau là các thông tin khác của class, tối thiểu các bạn hãy để tên tác giả theo mẫu trên. Làm đúng các việc trên, khi các bạn rê chuột vào tân class, sẽ có một cửa sổ hiện ra như sau (chú ý là các bạn đang code bằng IDE):

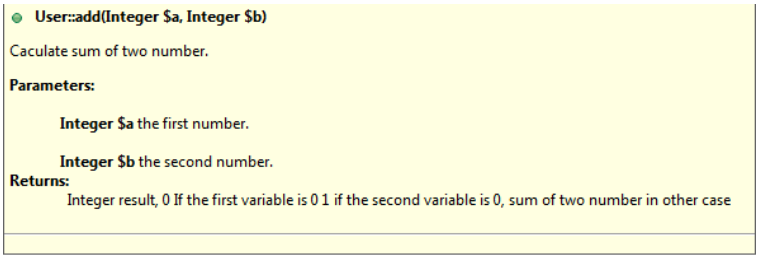


Document class theo đúng chuẩn

- Các hàm trong class cũng được thể hiện một cách tương tự, ngoại trừ có phần giải thích thêm cho các thông số truyền vào.

Bạn hãy đọc hàm sau đây và cảm nhận sẽ dễ hiểu như thế nào:

```
/**
 * Caculate sum of two number.
 * @param Integer $a the first number.
 * @param Integer $b the second number.
 * @return  Integer result, 0 If the first variable is 0 1 if the second variable is 0, sum of
 * two number in other case.
 *
 */
public function add($a,$b){
	if ($a==0){
		return 0;
	}
	else {
		if ($b==0){
			return 1;
		}
		else
			return $a+$b;
	}
}
```



Document hàm theo đúng chuẩn

c) Với biến và hằng

Document chức năng của mọi biến, hằng trước khi khai báo

Đây là điều rất cần thiết, nhất là với người đọc code. Ví dụ, chúng ta cần khai báo các thuộc tính tên, tuổi và nghề nghiệp cho class user, đồng thời khai báo một hằng tên công ty. Chúng ta sẽ thực hiện như sau:

```
//Company name, same for all user.
const COMPANY_NAME = "99xu";

//User's name
private $name;

//User's age
private $age;

//User's job
private $job;
```

d) Với các dòng code

Ghi chú càng rõ ràng, dễ hiểu càng tốt cho các dòng code

Các dòng xử lí lặp, rẽ nhánh, giải quyết vấn đề… đều phải được giải thích rõ ràng.

Hãy xem phiên bản hàm add được trình bày ở trên, với các lời giải thích rõ ràng:

```
public function add($a, $b) {
        // If the first parameter is 0, return 0.
        if ($a == 0) {

                return 0;
        } else {
                // If the second parameter is 0, return 1.
                if ($b == 0) {
                        return 1;
                }                       // Return sum of two parameters in other case.
                else
                        return $a + $b;
        }
}
```

IV. FORMAT CODE

Ở khía cạnh nào đó, thì một file mã nguồn cũng là một văn bản thuần túy, và hiển nhiên, vẫn có những cách format mà chúng ta phải tuân thủ. Một trong những qui tắc có thể kể ra như sau:

- Vị trí hai kí tự mở hàm ({) và kết thúc hàm (}) tương ứng.
- Giữa biến và phép toán phải có một khoảng chắn (Ví dụ: *$a == $b* thay vì **$a==$b**).
- Chữ cái đầu tiên của câu comment cách kí tự comment một khoảng chắn.

…

Tất nhiên, còn khá nhiều qui tắc nhỏ nhặt khác, và chắc chắn sẽ rất kinh khủng và mất thời gian nếu mỗi người lập trình viên phải tuân theo các qui tắc này.

Tuy nhiên, bạn không cần phải nhớ, và cũng không cần phải làm, vì các IDE sẽ làm giúp bạn.

Chúng ta có qui tắc cuối cùng:

Sau khi code xong và thực hiện tất cả các qui tắc trên, format code bằng IDE

Cách format code sẽ khác nhau tùy vào IDE, ví dụ, trên các IDE có nhân Eclipse sẽ là *Ctrl + Shift + F*, trên Netbeans là *Alt + Shift + F*.

Hãy xem sự khác biệt của hàm add hoàn chỉnh sau khi format code:

```
/**
        * Caculate sum of two number.
        *
        * @param Integer $a
        *            the first number.
        * @param Integer $b
        *            the second number.
        * @return Integer result, 0 If the first variable is 0 1 if the second
        *         variable is 0, sum of two number in other case
        *
        */
       public function add($a, $b) {
              // If the first parameter is 0, return 0.
              if ($a == 0) {

                     return 0;
              } else {
                     // If the second parameter is 0, return 1.
                     if ($b == 0) {
                            return 1;
                     }                    // Return sum of two parameters in other
case.
                     else
                            return $a + $b;
```

```
            }
}
```

Có thể thấy được code đã rõ ràng và dễ đọc hơn nhiều, nhất là ở phần ghi chú (documentation) ở đầu hàm.

V. BẢNG TỔNG HỢP QUI TẮC

Cuối cùng của tài liệu này sẽ là bảng tổng hợp các qui tắc khi lập trình trong hệ thống 99xu, rất mong mọi người sẽ thực hiện tốt các qui tắc này.

1. Mọi cái tên phải bằng tiếng anh, đúng chính tả và có nghĩa.
2. Đặt tên (biến, hàm, class, file…) theo qui tắc ở phần 2, mục II.
3. Document code bằng tiếng anh, đúng chính tả và càng rõ ràng càng tốt.
4. Mọi class và hàm phải document theo chuẩn
5. Document cho mọi biến, hằng trước mọi khai báo.
6. Ghi chú càng rõ ràng, dể hiểu càng tốt cho các dòng code
7. Sau khi thực hiện tất cả các điều trên và code hoàn tất, *format code* bằng IDE