Ministry of SMEs and Startups × KOSME Korea SMEs and Startups Agency × LIKELION

Coding Convention

Nguyen Anh Tuan

KTECH COLLEGE

Nội dung bài giảng

10 quy tắc lập trình

Java là gì?

Java Coding convention

Cài đặt và cấu hình môi trường Java

Cài đặt Spring Tool Suite

Viết ứng dụng Hello World

10 quy tắc lập trình

Keep It Simple, Stupid (KISS)

"Code càng đơn giản càng tốt". Nếu có thể viết kịch bản trong một dòng, hãy cứ viết kịch bản trong một dòng.

```
1   //Non KISS code
    no usages
2   function addNumbers(a:number, b:number):number{
3       return a+b
4   }
5
6   //KISS code
    no usages
7   const addNumbers2 = (a:number, b:number) => a+b;
```

Don’t Repeat Yourself (DRY Code)

“Không lặp lại code”. Cần sử dụng thuật toán hoặc hàm common để xử lý những đoạn code trùng lặp nhau.

```typescript
const teachers: string[] = ["Mr Hudson", "Mr James Kupelo", "Mrs Rainer White"];
const courses: string[] = ["Maths", "English", "Science"];

// Non-DRY code
console.log(teachers[0]);
console.log(teachers[1]);
console.log(teachers[2]);
console.log(courses[0]);
console.log(courses[1]);
console.log(courses[2]);

// DRY code

1+ usages
const logItems = (items: string[]): void => {
    for (let i : number  = 0; i < items.length; i++) {
        console.log(items[i]);
    }
};

logItems(teachers);
logItems(courses);
```

Open/Close

Các thư viện cần được mở rộng, nhưng không cho phép chỉnh sửa. Điều này sẽ giúp cho việc nâng cấp phiên bản trở nên rất dễ dàng, mà những người sử dụng sẽ không cần phải đập đi xây lại do đã chỉnh sửa vào bản trước đó.

```
1  class Shape {draw() : void {}}
2
   1+ usages
3  interface Scalable {
4      scale(factor:number):void;
5  }
6
   no usages
7  class ScaledShape extends Shape implements Scalable {
       no usages
8      scale(_factor: number) : void  {}
9  }
```

Composition Over Inheritance

Ưu tiên hợp đối tượng (object composition) thay vì thừa kế lớp (class inheritance). Các đối tượng có các hành vi phức tạp nên chứa các thể hiện của các đối tượng có các hành vi riêng lẻ. Chúng không nên kế thừa một lớp và thêm các hành vi mới.

```
1   //Composition
    1+ usages
2   class Job {
3       // variables, methods, etc.
4   }
5
    no usages
6   class Person {
7       // Composition (has-a relationship)
8       private job: Job;
9
10      // variables, methods, constructors, etc. object-oriented
11  }
12
13  // Inheritance
    1 inheritor  1+ usages
14  class Animal {
15      // variables, methods, etc.
16  }
17
    no usages
18  class Cat extends Animal {
19      // Additional variables, methods, etc. specific to Cat
20  }
21
```

Single Responsibility

Đơn nhiệm hay Trách nhiệm duy nhất có nghĩa chia nhỏ mỗi chức năng thành một class và các module nhỏ hơn. Điều này giúp cô lập một mô-đun nhất định để khắc phục sự cố dễ dàng hơn.

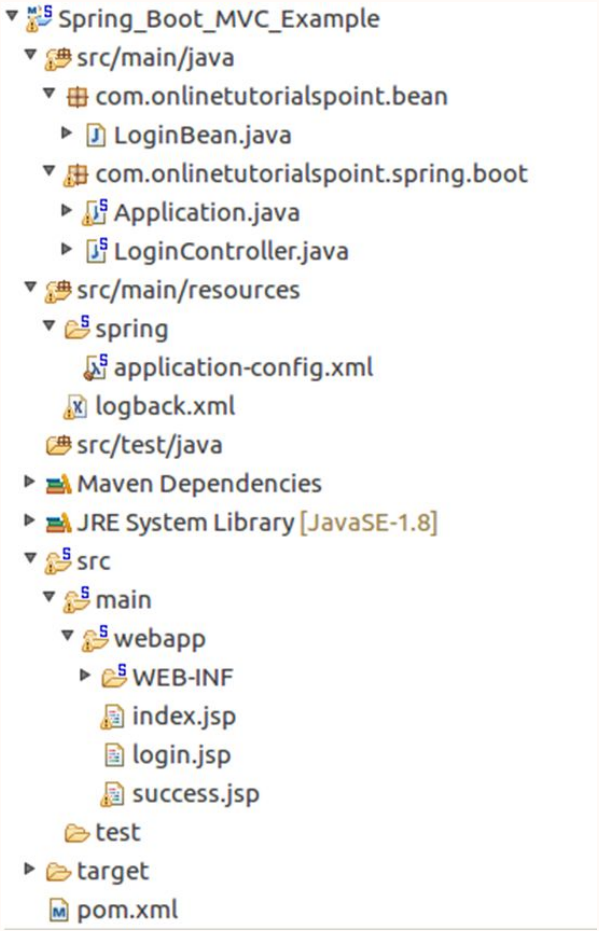
```
class Book {
    public title: string;
    public author: string;
    public description: string;
    public pages: number;

    //constructor and other methods
}

no usages
class Persistence {
    no usages
    public saveToFile(book:Book):void{
        // some fs.write method to save book to file
    }
}
```

Separation of Concerns

Một chương trình nên được thiết kế với các vùng chứa khác nhau. Mô hình thiết kế nổi tiếng MVC là một điển hình với ba vùng riêng biệt: Model - View – Controller.



You Aren’t Going to Need It (YAGNI)

Không nên cố gắng giải quyết một vấn đề không tồn tại. Thực tế có rất nhiều lớp trừu tượng, hàm common, thư viện được viết trong mã nguồn nhưng chưa bao giờ được sử dụng

```
class Author {
    firstName: string;
    lastName: string;
    1+ usages
    constructor(firstName: string, lastName: string) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    1+ usages
    getAuthorName(): string {
        return this.firstName + " " + this.lastName;
    }
}

no usages
class Program {
    no usages
    static main(): void {
        const author: Author = new Author("Joydip", "Kanjilal");
        console.log("Full name: " + author.getAuthorName());
    }
}
```

Document Your Code

Để lại ghi chú là một công việc mất thời gian hơn lập trình. Tuy vậy, không có gì là dùng một lần, ghi chú là rất quan trọng cho người cộng tác hoặc người kế nhiệm.

```typescript
// Represents a person with a name.
1+ usages
class Person {
    /**
     * Creates a new instance of the Person class.
     * @param {string} firstName - The first name of the person
     * @param {string} lastName - The last name of the person
     */
    1+ usages
    constructor(private firstName: string, private lastName: string) {}

    /**
     * Gets the full name of the person.
     * @return {string} The full nam of the person
     * */
    1+ usages
    getFullName(): string {
        return this.firstName + " " + this.lastName;
    }
}

// Example usage
const person: Person = new Person("John", "Doe");
console.log('Full name: ' + person.getFullName());
```

Refactor

Cấu trúc lại code có nghĩa là xem lại code và tìm cách tối ưu hóa nó, làm cho nó hiệu quả hơn trong khi vẫn giữ nguyên kết quả.

```
1  //Non refactor
   no usages
2  class Calculator {
       1+ usages
3      public add(a: number, b: number): number {return a + b;}
       1+ usages
4      public subtract(a: number, b: number): number {return a - b;}
       1+ usages
5      public multiply(a: number, b: number): number {return a * b;}
       1+ usages
6      public divide(a: number, b: number): number {return a / b;}
7
8  }
9  // Refactor
   no usages
10 class Calculator2 {
       no usages
11     public operate(a: number, b: number, operator: string): number {
12         switch (operator) {
13             case 'add': return a + b;
14             case 'subtract': return a - b;
15             case 'multiply': return a * b;
16             case 'divide': return a / b;
17             default: throw new Error('Invalid operator');
18         }
19     }
20 }
```

Clean Code

Hãy quên cái tôi đi! Đừng cố gắng đóng gói hàng tấn logic trong một dòng, điều đó không thể hiện sự thông minh của một lập trình viên giỏi.

```
1   //Non clean
    no usages
2   function calculateTotal(items: number[]):number{
3       let total : number  = 0;
4       for (let i : number  = 0 ; i < items.length; i++){
5           total += items[i]
6       }
7       return total
8   }
9
10  //Clean
    no usages
11  const calculateTotal2 = (items: number[]) =>{
12      return items.reduce((total : number  , currentItem : number ) => total + currentItem);
13  }
```

Java là gì?

Lịch sử phát triển

1. 1991: Sun Microsystem cho ra đời ngôn ngữ lập trình **Oak**
2. 1995: Oak đổi tên thành **Java**, lấy theo tên của hòn đảo trồng cafe tại Indonesia. Logo ly cafe của Java cũng xuất phát từ đây
3. 2010: Java được Oracle mua lại

Đặc điểm của Java

1. **Write once, Run anywhere:** Viết 1 lần, chạy mọi nơi, bất kể nền tảng như Windows, MacOS, Linux, …
2. Phân biệt với **Learn once, Write anywhere** của ngôn ngữ lập trình React
3. Đặc điểm nổi bật
    a. Hướng đối tượng
    b. Chạy trên mọi nền tảng
    c. Bảo mật cao
    d. Đa luồng
4. Sản phẩm của Java là: Mobile app, game, webapp, embedded devicces, …

Java Coding convention

Tiêu chuẩn Java Coding Convention

1. Là bộ quy tắc quy định cách viết code dành riêng cho Java và được viết bởi Oracle
2. Bao gồm:
    a. Đặt tên lớp, Interface, tên biến, phương thức, …
    b. Khoảng space, tab
    c. Khai báo, sử dụng biến
    d. Comment code
    e. Độ dài tối đa mỗi dòng, mỗi function, mỗi file, …

Tầm quan trọng của Coding Convention

1. Dễ bảo trì, sửa lỗi
2. Dễ đọc lại code
3. Làm việc nhóm hiệu quả hơn
4. Tiết kiệm thời gian, tối ưu hiệu suất
5. Có thể tái sử dụng trong các dự án khác nhau

Cài đặt và cấu hình Java

Giới thiệu 1 số phiên bản Java

1. JDK (Java Development Kit) là bộ công cụ cung cấp môi trường phát triển để viết và chạy ứng dụng bằng ngôn ngữ Java. bao gồm môi trường thực thi JRE (Java Runtime Environment), máy ảo JVM (Java Virtual Machine), và các công cụ, thư viện để phát triển ứng dụng
2. Lưu ý cài đặt phiên bản LST (Long term support) như : 8, 11, 17, 21

Cài đặt và cấu hình môi trường Java

1. Link tải JDK 11
   https://www.oracle.com/java/technologies/downloads/archive/
2. Cấu hình môi trường Java
3. Kiểm tra version bằng CMD: java --version

Cài đặt Spring Tool Suite

Giới thiệu 1 số IDE phổ biến

1. IDE là công cụ hỗ trợ việc viết code và chạy ứng dụng một cách dễ dàng hơn
2. Một số IDE phổ biến:
    a. Eclipse
    b. NetBeans
    c. IntelliJ
    d. JCreator
    e. Notepad

Download và cài đặt Spring Tool Suite

1. Link tải (Spring Tools 4 for Eclipse): https://spring.io/tools

Viết ứng dụng Hello World

Download và cài đặt Spring Tool Suite

Bước 1: Tạo project

Bước 2: Cấu hình project (Project name, JRE)

Bước 3: Tạo file main

Bước 4: In ra màn hình console dòng: “Hello World!”

Bước 5: Chạy chương trình

Lưu ý: Không sử dụng Auto Format Code (ctrl + shift + F)

ROAD TO KOREA

Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

mail@mail.com hoặc Zalo 0xxx xxx xxx