코딩 스타일 가이드

작성일자 : 2024년 4월 11일

작성자 : 묵‘Story

개정 이력

일자	개정 내용	작성자
2024년 4월 11일	코딩 스타일 가이드 최초 재정	묵'Story

[코딩 스타일 가이드]

1. 개요

- 많은 개발자들이 협업을 통해 애플리케이션을 개발 및 유지보수하고 운영합니다.
- 개발자별로 애플리케이션을 개발할 때 각자의 소스코드 작성 방식으로 개발하게 되는데
  이는 유지보수와 운영을 할 때 효율을 저하시키거나 오류를 발생시킬 수 있습니다.
- 이러한 문제를 사전에 예방하기 위해 개발자들이 공통적으로 지켜야 할 코딩 스타일을 제시합니다.
- 코딩 스타일 가이드에는 띄어쓰기 방식과 클래스/메소드/변수 명명 규칙, 그리고 클래스/메소드 등
  소스코드 작성 규칙을 제시하고 있으며
- 개발자들의 소스코드 작성 방식을 통일하여 애플리케이션의 소스코드를 일관되게 유지할 수 있도록
  돕습니다.
- 또한 애플리케이션을 효율적으로 유지보수 및 운영할 수 있도록 돕습니다.

2. 띄어쓰기 방식

- 모든 단어(클래스명, 메소드명, 상수명, 변수명 등)와 연산자들은 한칸의 띄어쓰기로 구분한다.

(예시)

```
// Method
public UserDto getUser(int id) {
    ...
}

// 변수 선언 및 객체 생성
UserDto userDto = new UserDto();
```

- 소스코드 작성 시 모든 들여쓰기는 한 번의 탭(Tab)으로 통일한다.

3. 명명 규칙

3.1. 클래스 명명 규칙

- 클래스의 이름은 알파벳 대문자로 시작합니다.
- 클래스의 이름은 카멜(Carmel) 표기법으로 작성합니다.
- 클래스의 이름은 명사로 구성합니다.

(예시)

```
// class
public class UserController {
    ...
}
```

3.2. 메소드 명명 규칙

- 메소드의 이름은 알파벳 소문자 또는 언더바(_)로 시작합니다.
  - 메소드의 접근자가 private인 경우 메소드의 이름을 언더바(_)로 시작합니다.
  - 메소드의 접근자가 private이 아닌 경우 메소드의 이름은 알파벳 소문자로 시작합니다.
- 메소드의 이름은 카멜(Carmel) 표기법으로 작성합니다.

- 메소드의 이름은 동사로 시작합니다.
  - 비즈니스 로직에서 메소드 작성 시 정보를 조회할 경우 get으로 시작합니다.
    - 목록을 조회할 경우 get 뒤에 단어를 복수형으로 작성합니다.
  - 비즈니스 로직에서 메소드 작성 시 정보를 저장할 경우 save로 시작합니다.
  - 비즈니스 로직에서 메소드 작성 시 정보를 수정할 경우 update로 시작합니다.
  - 비즈니스 로직에서 메소드 작성 시 정보를 삭제할 경우 remove로 시작합니다.
  - 데이터 접근 로직에서 메소드 작성 시 정보를 조회할 경우 select로 시작합니다.
    - 목록을 조회할 경우 select 뒤에 단어를 복수형으로 작성합니다.
  - 데이터 접근 로직에서 메소드 작성 시 정보를 저장할 경우 insert로 시작합니다.
  - 데이터 접근 로직에서 메소드 작성 시 정보를 수정할 경우 update로 시작합니다.
  - 데이터 접근 로직에서 메소드 작성 시 정보를 삭제할 경우 delete로 시작합니다.

(예시)

```
// public method
public List<UserDto> getUsers(String gender) { ... }

// private method
private String _getUserId(int id) { ... }

// DAO method
private List<UserDto> _selectUsers(String gender) { ... }
```

3.3. 상수 및 변수 명명 규칙

- 상수의 이름은 알파벳 대문자로 작성합니다.
- 상수의 이름은 스네이크(Snake) 표기법으로 작성합니다.
  - 상수 이름에 사용되는 단어 사이에는 언더바(_)로 구분합니다.
- 변수의 이름은 알파벳 소문자 또는 언더바(_)로 시작합니다.
  - 변수의 접근자가 private인 경우 메소드의 이름은 언더바(_)로 시작합니다.
  - 변수의 접근자가 private이 아닌 경우 메소드의 이름은 알파벳 소문자로 시작합니다.
- 변수의 이름은 카멜(Carmel) 표기법으로 작성합니다.
- 논리형(boolean) 변수는 is로 시작합니다.

(예시)

```
// 상수
public static final String HOME_PATH = “home”;

// public 변수
public String userId = “custUsr001”;

// private 변수
private String _gender = “MAN”;
```

4. 소스코드 작성 규칙

4.1. 공통으로 사용하는 정보(Constants, Enum 등)에 대한 소스코드 작성 규칙

- 어느 소스코드에서든 호출할 수 있도록 접근제어자를 public static으로 작성합니다.
- 다른 소스코드에서 수정할 수 없도록 final로 작성합니다.
- 페이지 경로, 메시지 내용 등 여러 소스코드에서 공통적으로 사용할 수 있는 정보는 별도의 파일로 작성합니다.
  - 예시) PageConstants, MessageConstants
- 페이지 경로와 관련된 Constants는 각 경로 단위로 작성한 후 조립하여 사용합니다.

(예시)

```
public static final String USER = “user”;
public static final String HOME = “home”;
// 최상위 경로
public static final String ROOT_PATH = this.USER;
// Home 페이지 경로
public static final String HOME_PATH = this.ROOT_PATH + this.HOME;
```

- RequestMapping에 사용되는 URL도 Constatns로 관리합니다.
- 공통 코드 등 여러 소스코드에서 공통적으로 사용할 수 있는 코드 정보는 별도의 Enum 파일로 작성합니다.
  - 예시) CommonCodeEnum

4.2. 공통 기능(Utils)에 대한 소스코드 작성 규칙

- 여러 소스코드에서 데이터 타입을 변경하거나 값이 있는지 체크하는 등 공통적으로 사용할 수 있는 기능은 별도의 파일로 작성합니다.
  - 예시) StringUtils, CryptoUtils, EscapeUtils

4.3. 클래스(Class)에 대한 소스코드 작성 규칙

- Controller, Service Dao
  - 하나의 Controller는 하나의 Service를, 하나의 Service는 하나의 Dao를 사용하도록 작성합니다.
    - 예시) AController에서 AService와 BService를 같이 선안할 수 없습니다.
      - 즉, AController에는 AService 또는 BService 중 하나만 선언할 수 있습니다.
    - 예시) AService에서 ADao와 BDao를 같이 선안할 수 없습니다.
      - 즉, AService에는 ADao 또는 BDao 중 하나만 선언할 수 있습니다.
- DTO
  - 전달되는 Data의 역살에 맞게 DTO를 작성합니다.
    - 예시) LoginDto, FindDto, UserDto, InfoDto, SearchDto 등
  - 로직을 통해 DTO의 값을 변경하지 않는다면 제어자(set) 메소드는 작성하지 않습니다.
  - 필요한 경우 일급객체를 활용하여 작성합니다.

4.4. 메소드(Method)에 대한 소스코드 작성 규칙

- 메소드의 파라미터는 3개 이하로 작성합니다.
  - 파라미터가 3개 이상 필요할 경우 DTO를 적절히 활용하여 작성합니다.
- 메소드의 로직 내에서 변수의 값을 수정하지 않는다면 파라미터를 final로 작성합니다.