

Ch2. 코딩 스타일

Jong-Hyeok Park
akindo19@gmail.com



바람직한 코딩 스타일의 기준

- 문서화 (주석)
- 코드 분할
- 명명 규칙
- 언어 사용
- 포매팅

코드 문서화

- 주식의 종류
 - 사용법
 - 알고리즘
 - 메타정보
 - Author, Date, Bug info

```
/*  
 * saveRecord()  
 * - 지정한 레코드를 데이터베이스에 저장한다.  
 * 리턴 값  
 * - int 저장된 레코드의 ID를 표현하는 정수  
 * 발생 가능한 exception  
 * - openDatabase() 호출하지 않는 경우  
 * DatabaseNotOpenException 발생함.  
 */
```

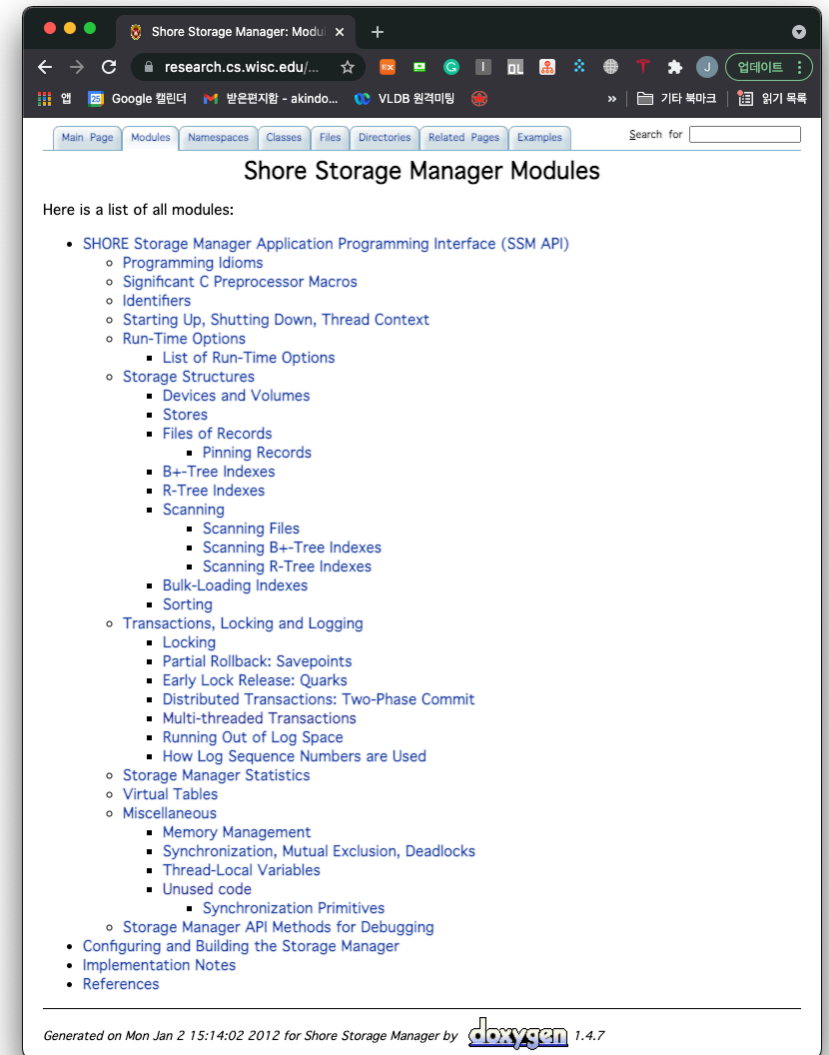
```
int saveRecord(Record& record);
```

```
/*  
 * 발생 가능한 exception:  
 * - openDatabase() 호출하지 않는 경우  
 * DatabaseNotOpenException 발생함.  
 */
```

```
RecordID saveRecord(Record& record);
```

코드 문서화

- 주석 스타일
 - 문장 단위
 - 머릿말
 - 고정양식 (doxygen)
 - 임의 주석



<https://research.cs.wisc.edu/shore-mt/onlinedoc/html/modules.html>

코드 문서화

- Doxygen (Cont.)
 - Shore-MT example

```
/**\brief Create a B+-Tree index.
 * \ingroup SSMBTREE
 * @param[in] vid Volume on which to create the index.
 * @param[in] ntype Type of index. Legitimate values are:
 * - t_btree : B+-Tree with duplicate keys allowed
 * - t_uni_btree : B+-Tree without duplicate keys
 * @param[in] property Logging level of store. Legitimate values are:
 * - t_regular
 * - t_load_file
 * - t_insert_file
 * See sm_store_property_t for details.
 * @param[in] key_desc Description of key type.
 * See \ref key_description for details.
 * @param[in] cc The locking protocol to use with this index. See
 * smlevel_0::concurrency_t and \ref SSMBTREE.
 * @param[out] stid New store ID will be returned here.
 */
static rc_t create_index(
    vid_t vid,
    ndx_t ntype,
    store_property_t property,
    const char* key_desc,
    concurrency_t cc,
    stid_t& stid
);
```

```
static rc_t ss_m::create_index ( vid_t      vid,
                                ndx_t      ntype,
                                store_property_t property,
                                const char * key_desc,
                                concurrency_t cc,
                                stid_t &    stid
                                )           [static, inherited]
```

Create a B+-Tree index.

Parameters:

[in]	vid	Volume on which to create the index.
[in]	ntype	Type of index. Legitimate values are: <ul style="list-style-type: none">• t_btree : B+-Tree with duplicate keys allowed• t_uni_btree : B+-Tree without duplicate keys
[in]	property	Logging level of store. Legitimate values are: <ul style="list-style-type: none">• t_regular• t_load_file• t_insert_file See sm_store_property_t for details.
[in]	key_desc	Description of key type. See Key Types for details.
[in]	cc	The locking protocol to use with this index. See smlevel_0::concurrency_t and B+-Tree Indexes .
[out]	stid	New store ID will be returned here.

코드 문서화

- 주석을 작성하기 전에 굳이 주석을 달 필요가 없도록 코드를 수정할 수 있는지 검토한다

코드 분할

- 리팩토링
 - 추상화 : 필드 캡슐화, 타입 일반화
 - 논리성 : 메서드, 클래스 추출
 - 명칭 및 위치 개선 : pull up , push down

코드 분할

- 디자인 기준
 - 코드작성 보다 디자인 먼저!
 - 코드 밀집도 ↓ 체계성 ↑
 - 가상 함수/클래스

base.hpp

```
class Car {  
public:  
    virtual showPrice() = 0;  
    virtual void Start() = 0;  
};
```

sportCar.hpp

```
class SportCar : public Car {  
    virtual void Start() {  
        std::cerr << "1000M$";  
    }  
    virtual void Start() {  
        std::cerr << "부아아아앙";  
    }  
};
```

miniCar.hpp

```
class MiniCar: public Car {  
    virtual void Start() {  
        std::cerr << "1$";  
    }  
    virtual void Start() {  
        std::cerr << "슈슈슈속";  
    }  
};
```


명명 규칙

- 컴파일러 규칙
 - 이름의 첫 글자 숫자 쓰면 안됨
 - “_” 특수용도로만 사용 가능함
 - “_” 로 시작하는 이름 사용 불가
 - 123var, __var, _var
- 카운터
 - 2D data 다루는 경우 row, column 표기 (직관적)
 - 중첩 반복문의 경우 Inner, Outer 표기

명명 규칙

- 접두어

접두어	예	본래단어	용도
m m_	mData m_data	member	클래스의 데이터 멤버
s ms ms_	sLookupTable msLookupTable ms_lookupTable	static	정전변수 또는 데이터 멤버
k	kMaxLength	konstant	상숫값
b is	bCompleted isCompleted	Boolean	부울값
b mNum	nLines mNumLines	number	카운터로 사용하는 데이터

명명 규칙

- Getter, Setter
- 대소문자 활용
- Namespace를 적용한 상수

레퍼런스

- 레퍼런스 사용의 장점
 - 안정성 : `nullptr(x)`, 주소 직접 다루지 않음.
 - 코딩 스타일 (불필요한 `*`, `&` 기호)
 - 메모리 소유권 명확히 표현 가능

포매팅

- 중괄호

```
void foo() {  
    if (condition)  
    {  
        /* ... */  
    } else {  
        /* ... */  
    }  
}
```

```
void foo()  
{  
    if (condition)  
    {  
        /* ... */  
    }  
    else  
    {  
        /* ... */  
    }  
}
```

```
void foo()  
{  
    if (contidion)  
        var = 1;  
  
    if (condition) {  
        var = 1;  
    }  
}
```

포매팅

- 스페이스
 - indent
 - expandtab (vim)
 - Makefile

```
if(i==2){
```

```
...  
}
```

```
if ( i==2 ) {
```

```
...  
}
```

```
if ( i == 2 ) {
```

```
...  
}
```

References

[1] Marc Gregoire, 2018, Professional C++, 4th edition, WILEY

[2] https://research.cs.wisc.edu/shore-mt/onlinedoc/html/d4/df6/group___s_s_m_b_t_r_e_e.html