**SMARTsuite9.x 개발 표준 정의서**

**VER 1.0**

**2020년2월**

**제,개정내역**

|  |  |  |  |
| --- | --- | --- | --- |
| 버전/릴리스 | 작성일자 | 작성자 | 개요 |
| 1.0 | 2020.2.20 | 김 광 섭 | 신규제정 |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| 목 차 |

[1. 개요 5](#_Toc34914432)

[**1.1 목적** 5](#_Toc34914433)

[**1.2 범위** 6](#_Toc34914434)

[2. 공통 6](#_Toc34914435)

[3. 개발표준 7](#_Toc34914436)

[**3.1 패키지 구조** 7](#_Toc34914437)

[**3.1.1 클래스 패키지 구조** 7](#_Toc34914438)

[**3.1.2 UI 구조** 8](#_Toc34914439)

[3.1.2.1 Html 구조 8](#_Toc34914440)

[3.1.2.2 Web Resource(js, css, img, ..)구조 8](#_Toc34914441)

[**3.2 파일명** 9](#_Toc34914442)

[**3.2.1 Java 파일명** 9](#_Toc34914443)

[**3.2.2 Query(mybatis) 파일명** 9](#_Toc34914444)

[**3.2.3 화면 파일명** 10](#_Toc34914445)

[**3.2.3.1 화면 ID** 10](#_Toc34914446)

[**3.3 서버(java) 소스** 12](#_Toc34914447)

[**3.3.1 SQL- Query ID** 12](#_Toc34914448)

[**3.3.2 메시지ID** 13](#_Toc34914449)

[**3.3.3 메소드 명** 13](#_Toc34914450)

[**3.3.4 객체명** 14](#_Toc34914451)

[**3.3.5 변수명** 14](#_Toc34914452)

[**3.3.6 상수명** 15](#_Toc34914453)

[**3.4 주석 (Comment)** 16](#_Toc34914454)

[**3.4.1 서버 주석** 16](#_Toc34914455)

[4. 개발환경 세팅 17](#_Toc34914456)

[**4.1 서버 개발환경** 17](#_Toc34914457)

[**4.1.1 이클립스** 17](#_Toc34914458)

1. **개요**
   1. **목적**

SMARTsuite 개발업무에 개발의 표준을 제시함으로써 개발자들의 생산성을 향상시키고, 유지보수를 용이하게 하며, 사용자들의 편의성을 도모하고, 시스템에 대한 일관성 유지로 사용자 요구사항에 신속 대응함을 목적으로 한다.

소프트웨어 개발 시 적용되는 프로그램 구조 명 (클래스, 인스턴스, 멤버상수 등) 과 이에 관련되는 화면명, 파일명 등에 대한 명명규칙(Naming Rule) 을 정의하여 개발 작업이 일관성 있고 정확하게 수행되도록 가이드 한다.

* 1. **범위**

프로젝트 내부에서 작성되는 모든 어플리케이션에 대해서 적용한다.

1. **공통**

* 모든 이름은 a-z, A-Z, 0-9 의 글자로 조합한다.
* 프로그램 내에서 사용하는 모든 이름은 64자를 넘지 않는 것으로 한다.
* 정확한 의미 전달을 위해 약어의 사용은 피한다.
* 약어 사용이 의미 전달에 도움이 되는 경우에는 용어사전(메타시스템)에 포함된 약어를 사용할 수 있으나, 그렇지 않은 경우에는 단어의 spelling 은 모두 적는 것으로 한다.
* 특수기호는 사용하지 않는다.
* 의미가 비슷한 이름이나, Case(대/소) 만 다른 이름은 사용하지 않는다.

1. **개발표준**
   1. **패키지 구조**
      1. **클래스 패키지 구조**

본 솔루션의 패키지 구조는 아래와 같은 규칙으로 작성한다.

* 모두 소문자로 작성한다.
* Level1 : 솔루션명이 첫번쩨 패키지 명으로 사용된다.

ex) smartsuite-app-bp-pro-pr일 경우 smartsuite

* Level2 : 솔루션명의 각 framework 레벨이 패키지 명으로 사용된다.

ex) smartsuite-app-bp-pro-pr일 경우 app (Application 영역)

* Level3 : 솔루션의 사용가 구분코드가 사용된다.

ex) smartsuite-app-bp-pro-pr일 경우 bp (bp : Buyer Portal, sp : Supplier Portal)

* Level4 : 솔루션의 대 업무 영역이 사용된다.

|  |  |
| --- | --- |
| **종류** | **설명** |
| **cms** | 품목관리 |
| **esourcing** | 협력사관리 |
| **pro** | Procument |
| **srm** | 협력사 평가 |
| **…** | … |

* Level5 : 솔루션의 세부 업무 영역이 사용되며 개발자가 구현해야 할 Controller, Service의 Java 파일들이 위치하는 Layer다.

ex) smartsuite-app-bp-pro-pr일 경우 pr (구매요청)

* Level6 : 각 Layer 별로 개발자가 구현해야 할 Java 파일들이 위치한다.

|  |  |
| --- | --- |
| **구분** | **설명** |
| **패키지 명명법** | Level-1 + ‘.‘ + Level-2 + ‘.‘ + Level-3 + ‘.‘ + Level-4 + Level-5 + Level-6 |
| **예시** | smartsuite.app.bp.pro.pr  smartsuite.app.bp.cms.item  smartsuite.app.bp.pro.pr.validator |

* + 1. **UI 구조**

#### Html 구조

smartsuite UI 소스 파일(html)은 각 프로젝트의 src/main폴더 하위에 작성한다

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **구분** | **Level 0** | **Level 1** | **Level 2** | **Level3** | **비고** |
| project | src/main | ui | cmd |  | 패키지의 level4에 해당하는 업무명을 사용 |
|  |  |  | item | 패키지의 level5에 해당하는 업무 명을 사용 |
|  |  | pro | ar |  |
|  |  |  | auction |  |
|  |  |  | pr |  |
|  |  |  | rfx |  |
|  |  |  | shared | level2에서 공통으로 사용하는 ui소스가 위치하는 폴더 |

#### Web Resource(js, css, img, ..)구조

smartsuite UI Web Resource(js, css, img, …)은 webapp폴더 하위에 작성한다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **구분** | **Level 0** | **Level 1** | **Level 2** | **Level3** | **비고** |
| webapp | css |  |  |  | css파일 |
| img |  |  |  | 이미지 파일 |
| js |  |  |  | javascript파일 |
| resources |  |  |  |  |
| WEB-INF |  |  |  | 웹 어플리케이션이 실행하는데 필요한 파일 및 설정파일 등이 저장되는 폴더 |
|  | jsp |  |  | Jsp파일 |

* 1. **파일명**
     1. **Java 파일명**

Controller도 각 업무 레벨에 위치하여 작성한다.

Service 는 Interface를 별도 작성하지 않는다.

Java 파일명은 SMARTsuite9.1\_codding-style.pptx파일을 참고한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **종류** | **Class 종류** | **postfix** | **명명규칙** | **예시 (사용자관리)** |
| Java 파일명 | Controller | Controller | CalssName\_Contoller | UserController.java |
| Service | Service | ClassName + Service | UserService.java |

* + 1. **Query(mybatis) 파일명**

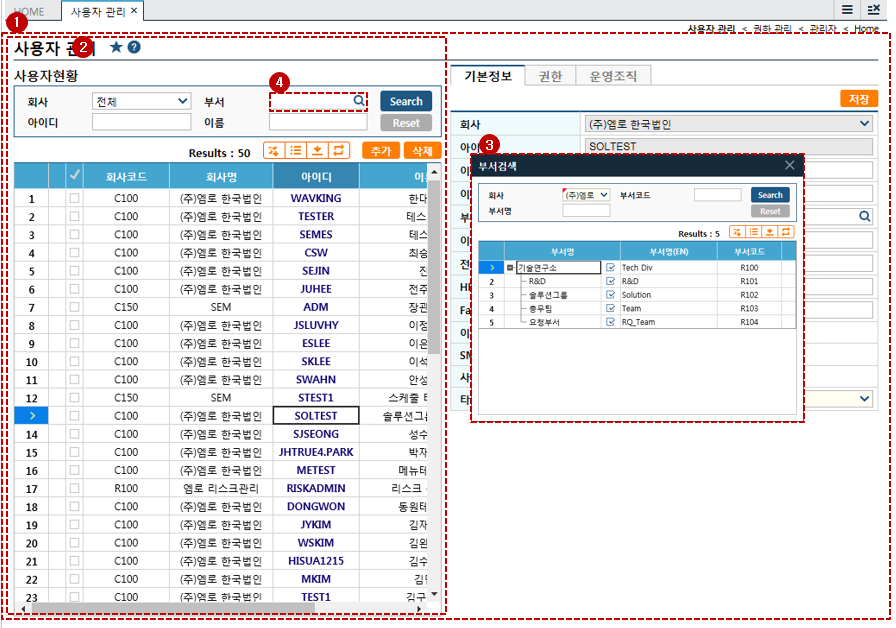
Service 단위로 생성하고 파일명은 **Service를 생략한 파일명**으로 명명한다.

* 파일의 확장자는 xml 이다.
* 파일의 단위는 Service 단위이며 Service 파일명에서 Service를 제외한 명칭이다.
* 파일명은 소문자로 작성하며 이어지는 단어의 구분자는 하이픈(-)을 사용한다.
* 개발 시 파일의 위치   
  /src/main/resources/smartsuite/mappers/**DBVendor별 폴더**/level2업무명/파일명.xml
  + 1. **화면 파일명** 
       1. **화면 ID**

화면은 유형별로 pre-fix를 붙여서 생성한다.

**\* cc-modume-behaivior에서 pre-fix로 자동화 처리를 하고 있기 때문에 준수하여야 함.**

* 화면의 메인 모듈 소스는 em(emro-module) pre-fix사용 -그림 1번
* 개별 프로세스화면 소스는 es(emro-submodule) pre-fix사용 -그림 2번
* 팝업 화면 소스는 ep(emro-popup) pre-fix 사용 -그림 3번
* 커스텀 컴포넌트를 만드는 경우 cc(custom-component) pre-fix 사용



* 1. **서버(java) 소스**
* SMARTsuite9.1\_codding-style.pptx파일을 참고한다..
  + 1. **SQL- Query ID**

Mybatis는 Service 와 동일한 명명 규칙으로 파일을 생성하며 xml 파일 하나에 하나 이상의 쿼리를 작성할 수 있고 각각의 쿼리 문장마다 고유 id 를 부여한다.

예) UserService.java, user.xml

user.xml

<select id=" findListUser" resultType="map">

SQL 쿼리 내용

</select>

기본적으로 아래와 같이 명명하며 테이블 종속적인 DBIO 는 기본적으로 생성해준다.

만약, 참조모델에서 Query ID 를 여러 개를 사용해야 한다면 "01" 과 같이 일련번호(2자리) 를 구분하여 사용한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **규칙** | | **예시** | **리턴** |
| 조회 | find + 자연어, findList + 자연어 | findItem findListItem | map,  Array |
| 조회 | get + 자연어 | getItemName | 단일타입 |
| 입력 | insert + 자연어 | insertItem |  |
| 수정 | update + 자연어 | updateItem |  |
| 삭제 | delete + 자연어 | deleteItem |  |
| 입력(수정) | merge + 자연어 | mergeItem |  |

* + 1. **메시지ID**
* 서버 프로그램 수행 시 발생되는 오류 종류와 내용을 코드화 하여 일관된 형태의 메시지를 표시하기 위해 메시지 처리 규칙을 정한다.
* ID와 유형은 대문자로 사용한다.

| * **종류** | * **구분** | * **규칙** |
| --- | --- | --- |
| 메시지ID | 규칙 | * 코드와 유형은 대문자로 사용한다. * **STD.level5업무코드+일련번호(4)** |

* + 1. **메소드 명**
* method 는 CRUD를 나타내는 동사와 업무를 나타내는 명사를 사용하여 정의하며 다음의 규칙을 준수한다.
* 단어 조합 시 다음과 같은 형태로 작성한다.
* 동사 + 명사의 구조로 명명한다.
* 단건/목록을 구분하기 위해 목록 처리의 경우 list 를 붙인다.
* "-", "\_", "$"은 사용하지 않음

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **작업내용** | **prefix** | **postfix** | **예시** | **적용범위** |
| 저장/수정 | save |  | saveUser | insert / update |
| 조회 | find |  | findUser | select |
| 삭제 | delete |  | deleteUser | delete |
| 조회(목록) | findList |  | findListUser | select |
| 정보얻기 | get |  | getUserName | select |

* + 1. **객체명**

클래스명과 동일하게 하는 것을 원칙으로 한다.

* 클래스명과 동일하게 사용하는 경우 -> UserService userService;
  + 1. **변수명**
* 변수명은 소문자를 기본으로 사용한다.
* 변수명이 두 단어 이상의 조합일 경우, 다음에 시작하는 단어 별로 첫 문자를 대문자로 표기한다. (lowerCamelCase)
* 변수 앞에 약어 형태의 접두사를 붙이는 경우에는 약어를 모두 소문자로 표기한다.

sqlDatabase, mapParam

* JSP/HTML/JavaScript에서 사용하는 폼변수, 자바코드에서 사용하는 변수, 데이터 베이스에서 사용하는 속성명(컬럼명)은 동일하게 부여하여 개발의 편의성을 확보한다. (단, 속성명(컬럼명)은 다른 이름과 달리 "\_"를 음절이 구별되어 표현되나, 폼변수 및 자바변수는 "\_"를 생략하고, 시작문자는 소문자로 시작하고 단어구분자는 대문자로 표현한다.)

예) DB 속성명(GMM\_DIC\_NM) --> JSP/HTML, 자바변수명(gmmDicNm)

* + 1. **상수명**
* 상수는 모두 대문자로 표기한다.

예) MESSAGE

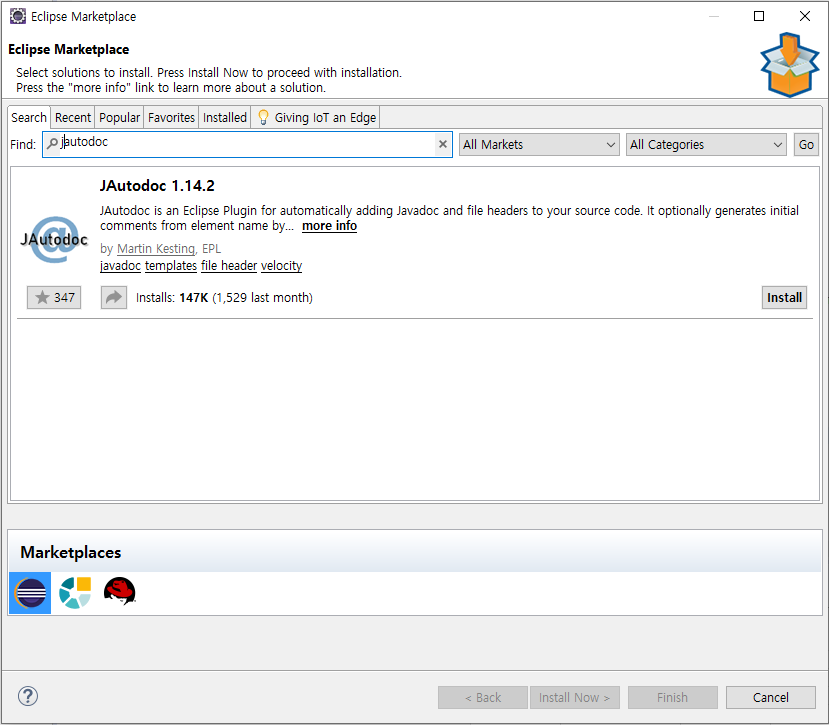
* 복합어일 경우 단어와 단어 사이를 "\_"로 구분한다.

예) ERROR\_MESSAGE

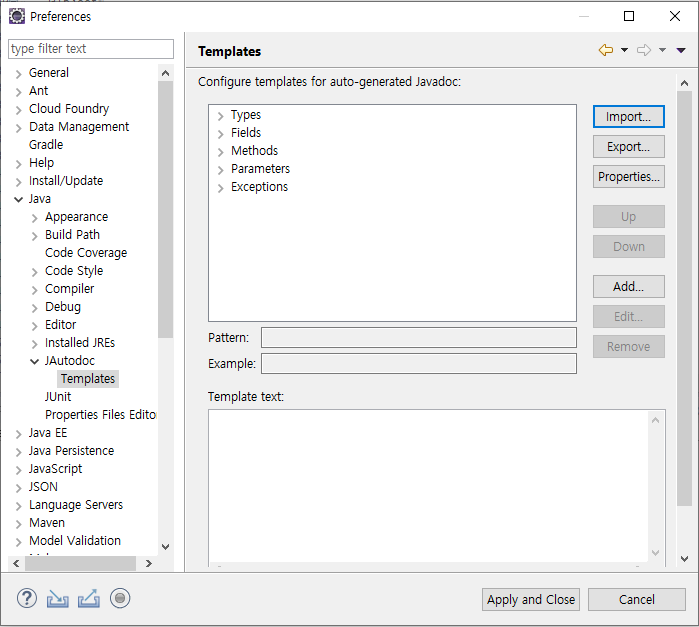
* 1. **주석 (Comment)**
     1. **서버 주석**

서버의 주석의 경우 개발도구에 포함된 Code Template 을 적용을 기본으로 작성한다.

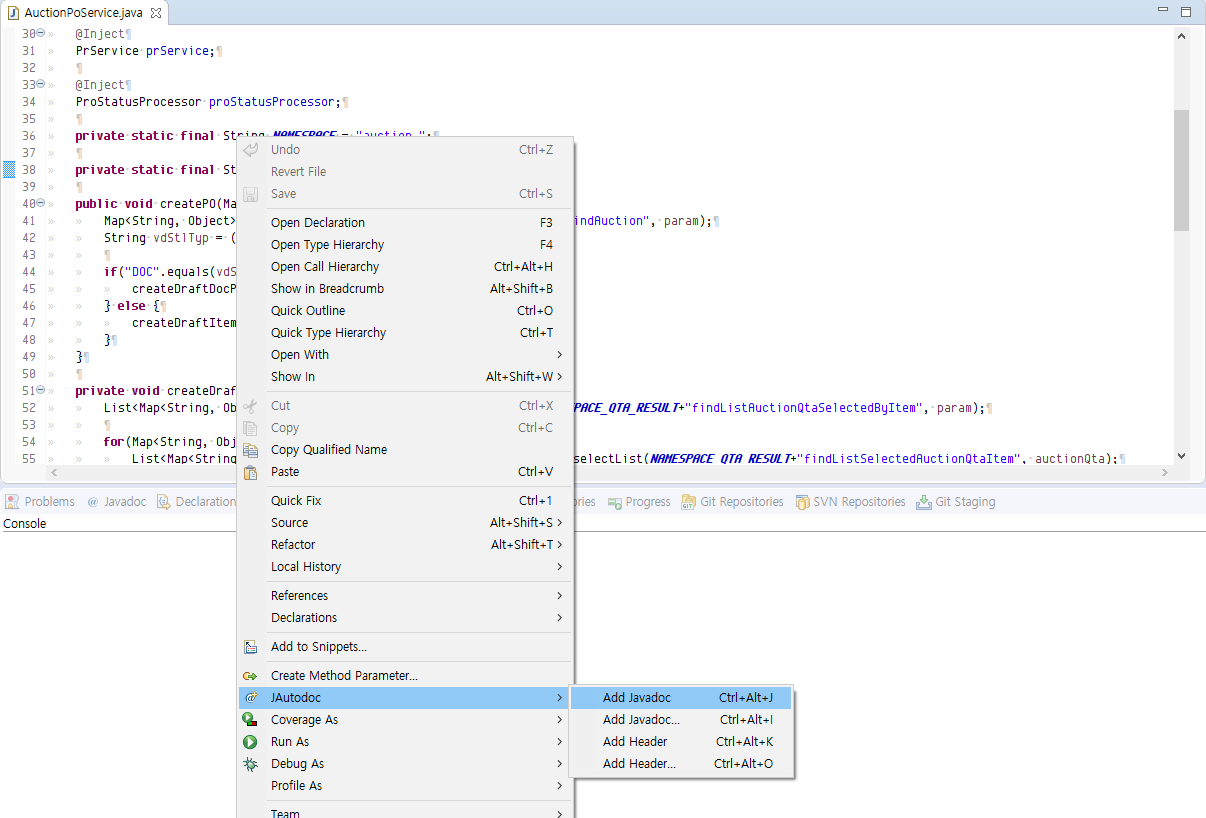
* Method명으로 의미를 알 수 있는 경우 주석을 생략한다.
* 다른 시스템과의 연계 및 개발자에게 알려야 하는 정보가 있는 경우 주석을 작성한다 예) param(operOrg : 운영단위 코드, rfxNo: RFx 번호 …)
* 함수, 변수에 대해 주석을 작성하되, 유지보수 및 코드이해를 위해 가급적 상세히 표시한다.
* 만일 사이트에서 전체 Class, Method에 대한 주석이 필요한 경우 eclipse의 Java Doc plug-in 을 사용하여 제공된 format으로 자동 생성되도록 한다.
  + 1. **Java Doc plig-in**



eclipse – help – Eclipse Marketplace 에서 JautoDoc을 검색하여 설치한다.



eclipse – window – preferences – java – JautoDoc – Templates 화면에서 제공한 xml파일을 Import한다



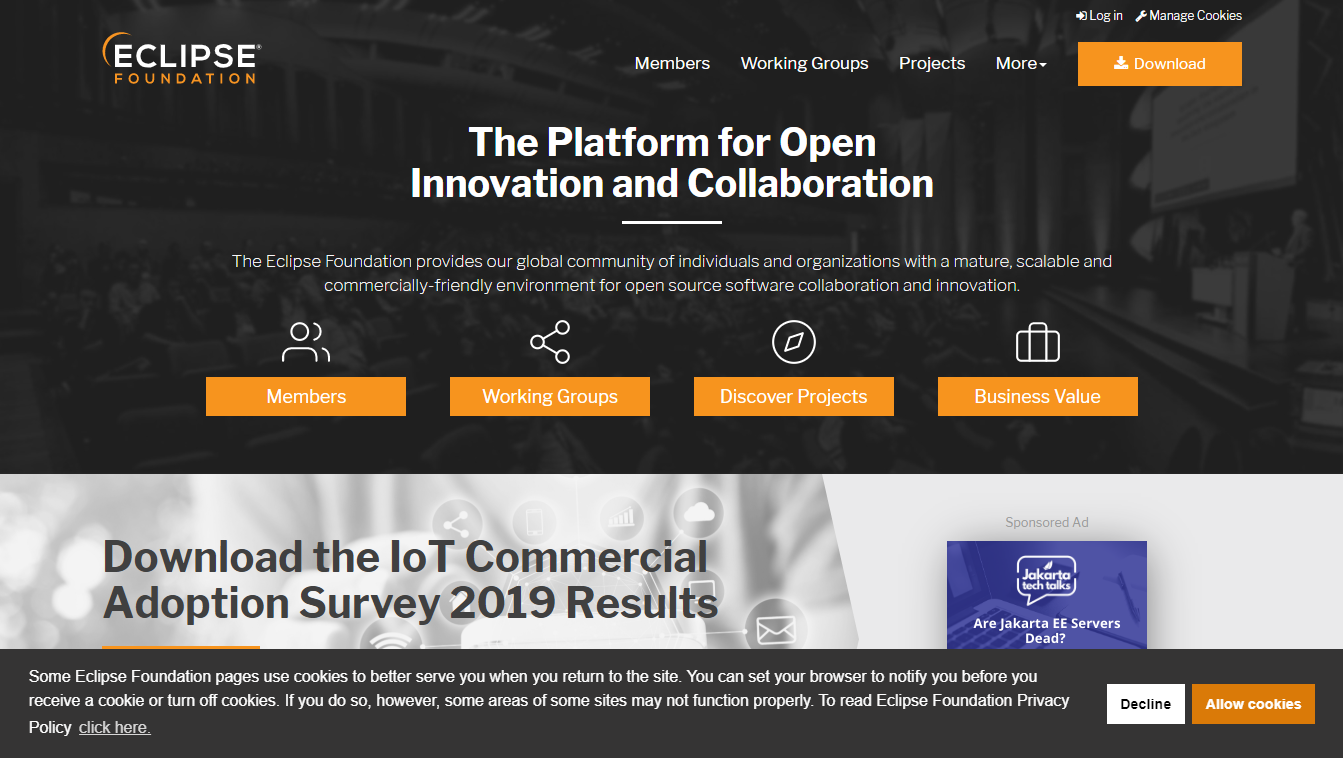
해당 Class에서 마우스 오른쪽 버튼 JAutodoc – Add Javadoc을 클릭하면 해당 Class 및 Method에 주석이 자동으로 입력된다.

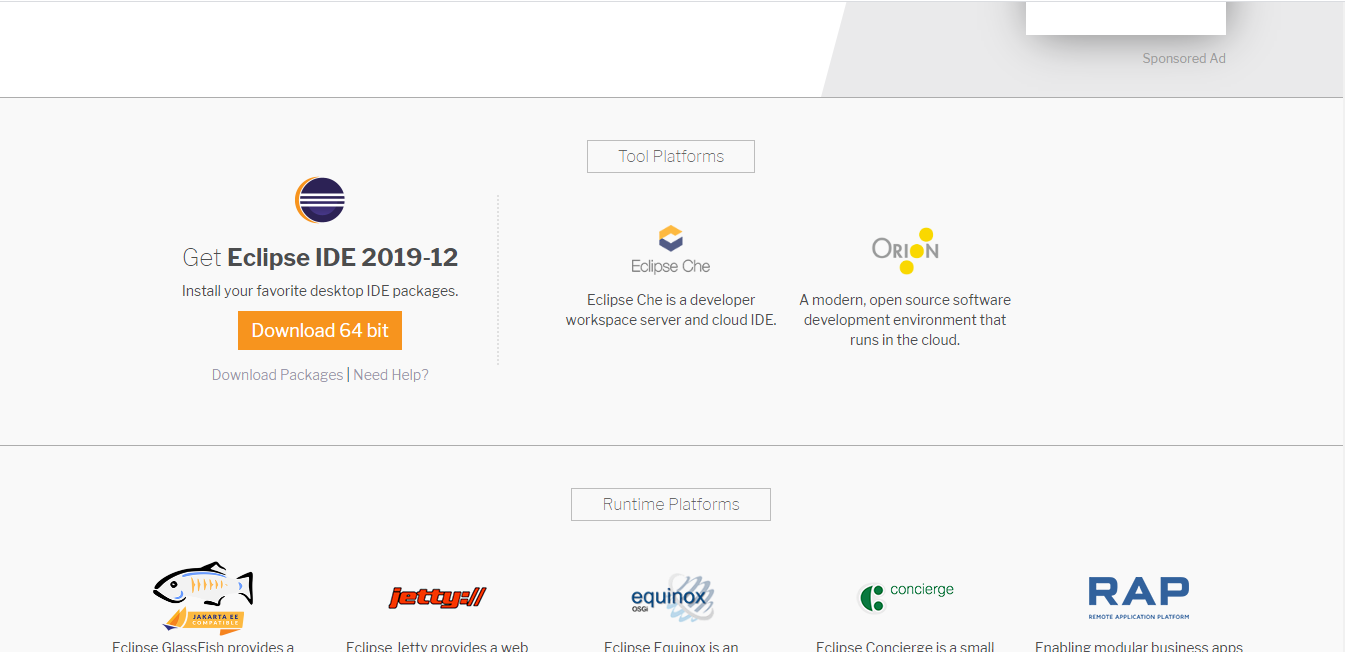


입력된 주석을 확인할 수 있다.

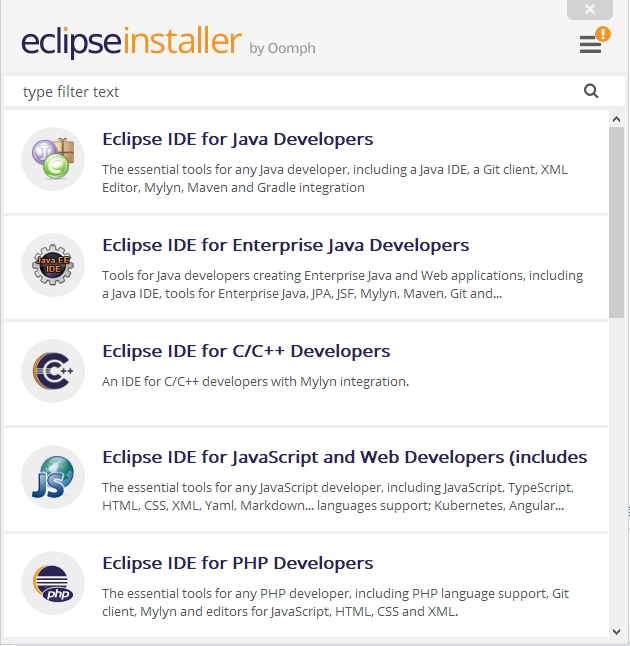
1. **개발환경 세팅**
   1. **서버 개발환경**
      1. **이클립스**

* eclipse.org 사이트에 접속하여 최신 버전의 eclipse를 다운받아 설치한다.

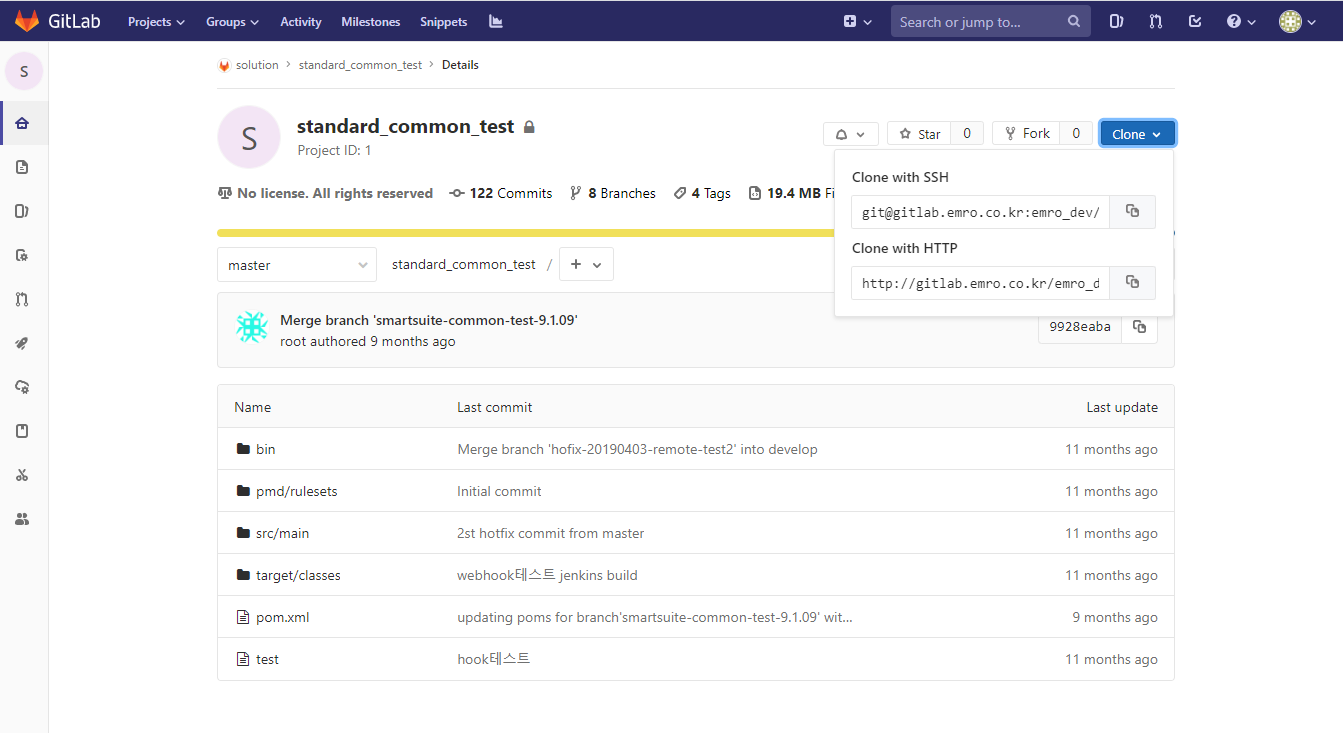




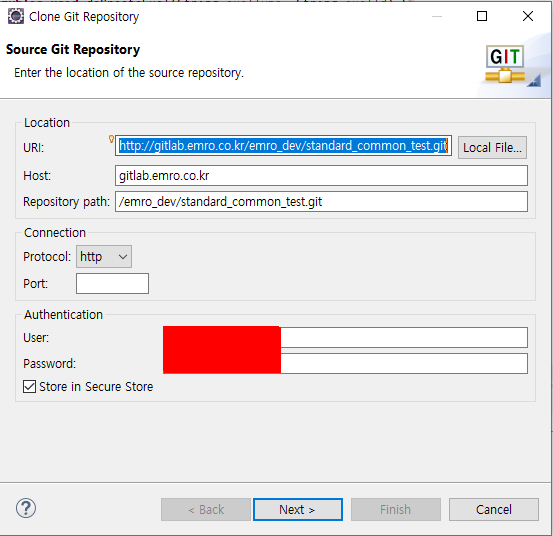
* eclipse installer에서 **Eclipse IDE for Enterprise Java Developers**를 선택하여 eclipse를 설치한다.



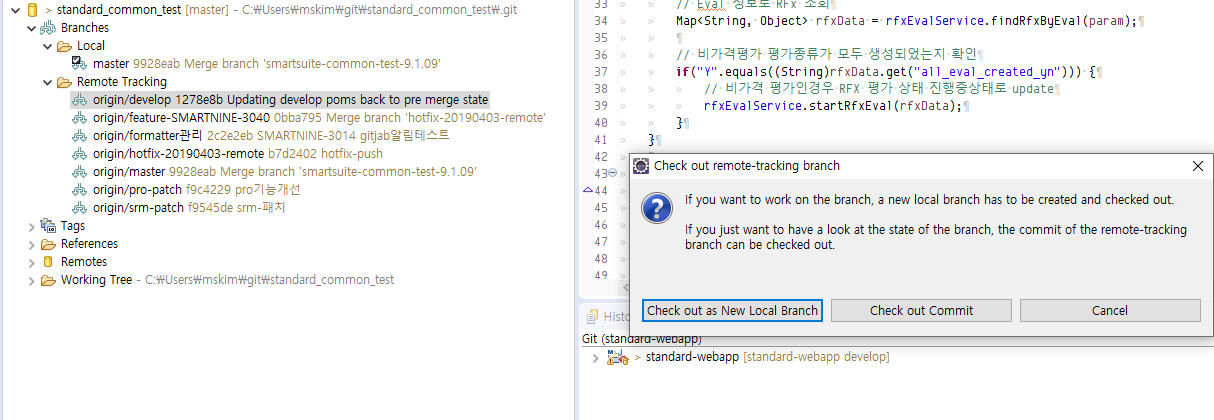
* 이클립스 실행 후 GitLab에서 개발할 프로젝트를 Clone 한다.
* GitLab URL: http://gitlab.emro.co.kr/
* Sample Project: http://gitlab.emro.co.kr/emro\_dev/standard\_common\_test.git



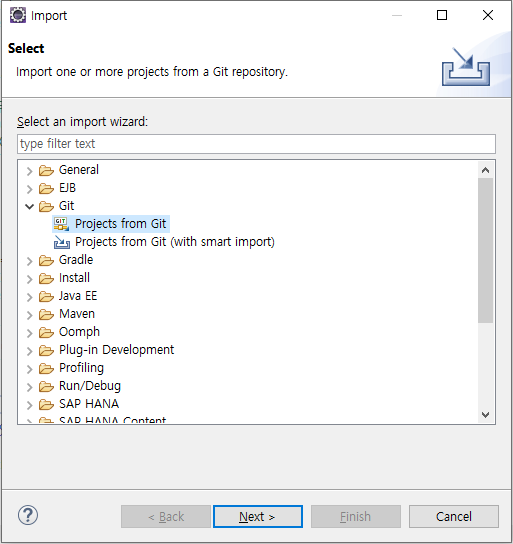
* Eclipse의 Git Repository View에서 프로젝트 우클릭 -> paste Repository Path or URI를 클릭하여 Repository를 등록한다.



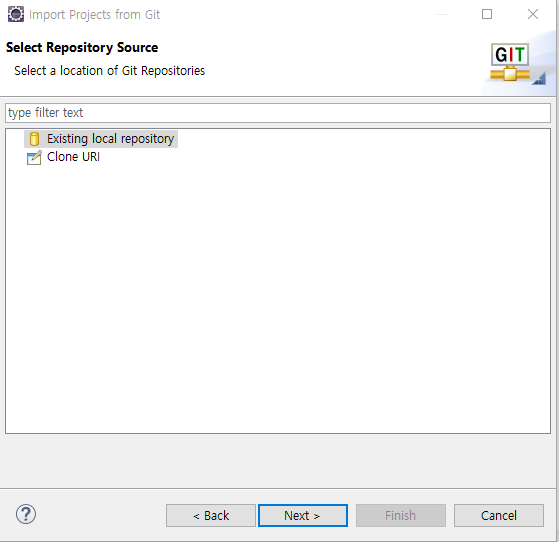
* 프로잭트 하위의 Branches – Remorte Tracking – origin/develop을 더블클릭하여 Local Branche를 생성하고 선택한다.

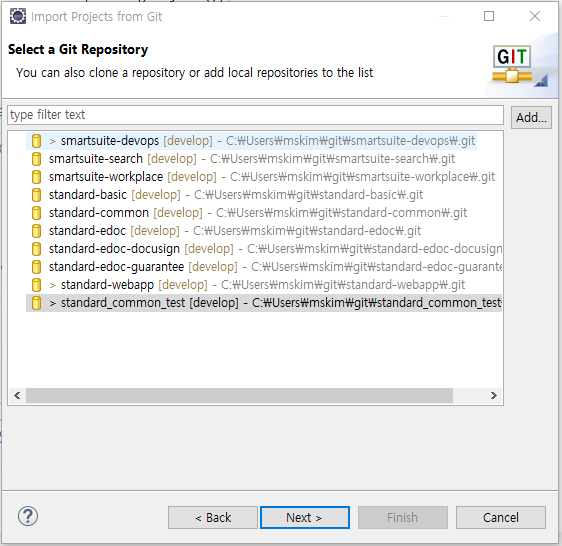


* Eclipse Explorer에서 마우스 우 클릭 -> import

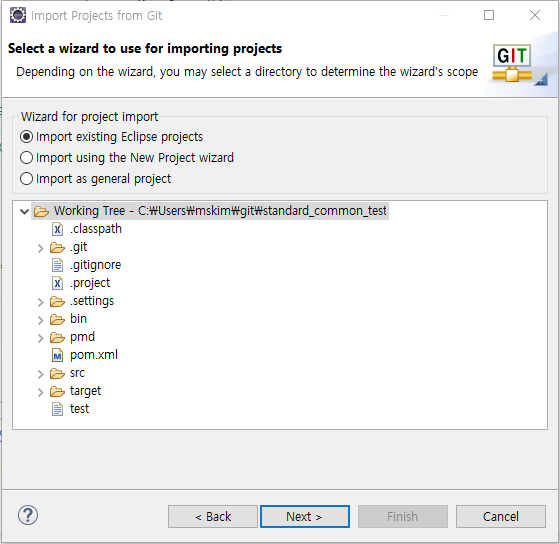


* Git – Projects from Git 선택
* 위의 내용처럼 repository가 연결되어 있는 경우 Existing local repository를 선택하고 위의 내용을 생략하는 경우 Clone URI를 선택하여 위의 내용처럼 진행함.

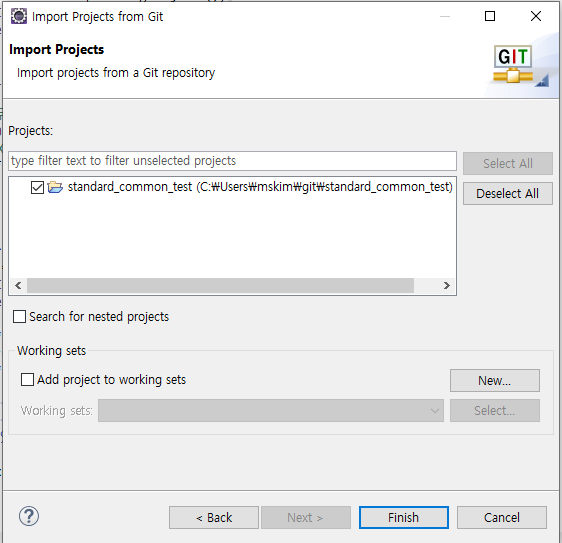




* 연결된 repository를 선택



* import existing Eclipse Projects를 선택하고 Next



* search for nested projects 선택해제 후 Finish버튼 클릭으로 프로젝트 생성
* 이후 Maven -> update Project를 실행
* Git을 잘 사용하지 못한다면 다음 링크를 통해 Git을 알아보고 공부 후 사용하도록 함
* Git 기초 : <http://techdocs.emro.co.kr/pages/viewpage.action?pageId=9517283>