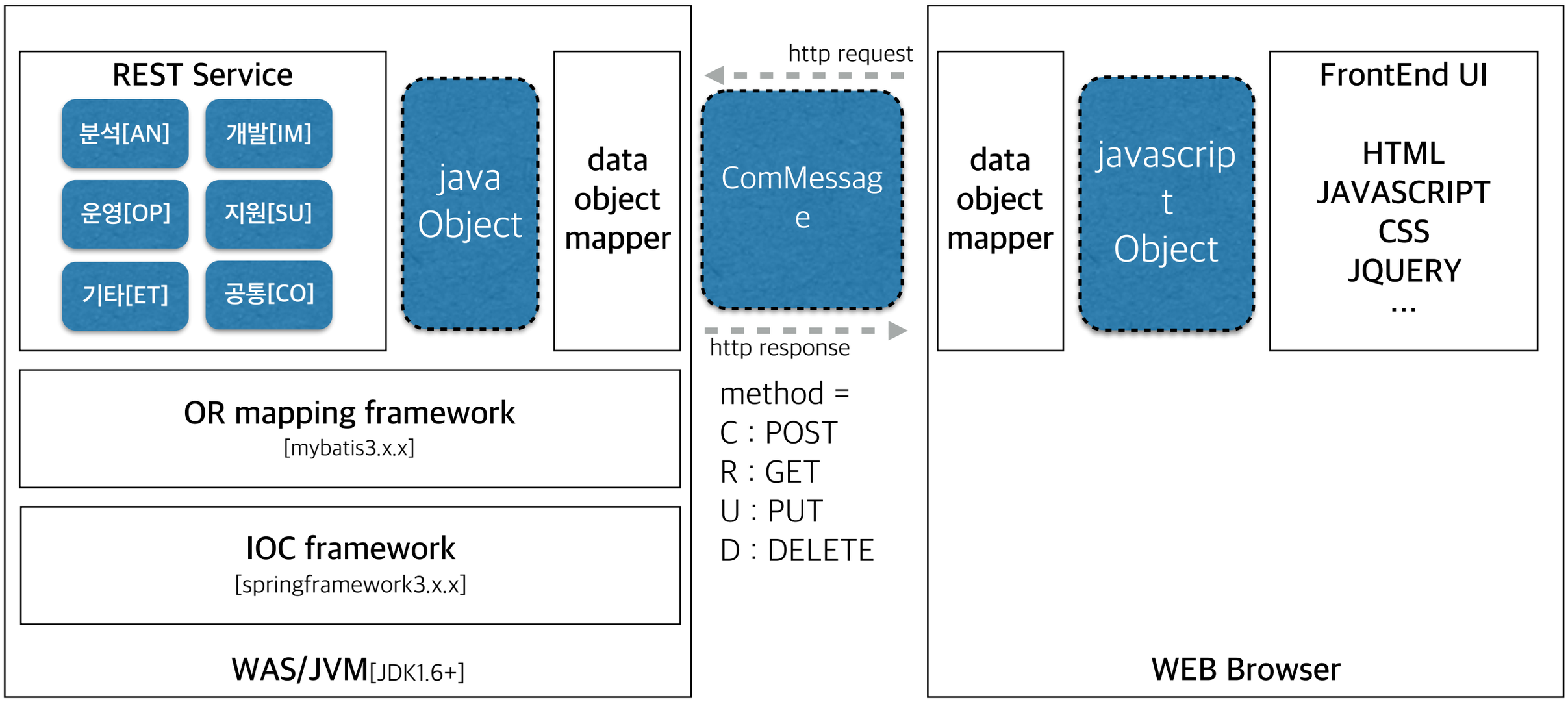
백엔드 개발 가이드

1.REST Service 개발 절차

REST Service Architecture

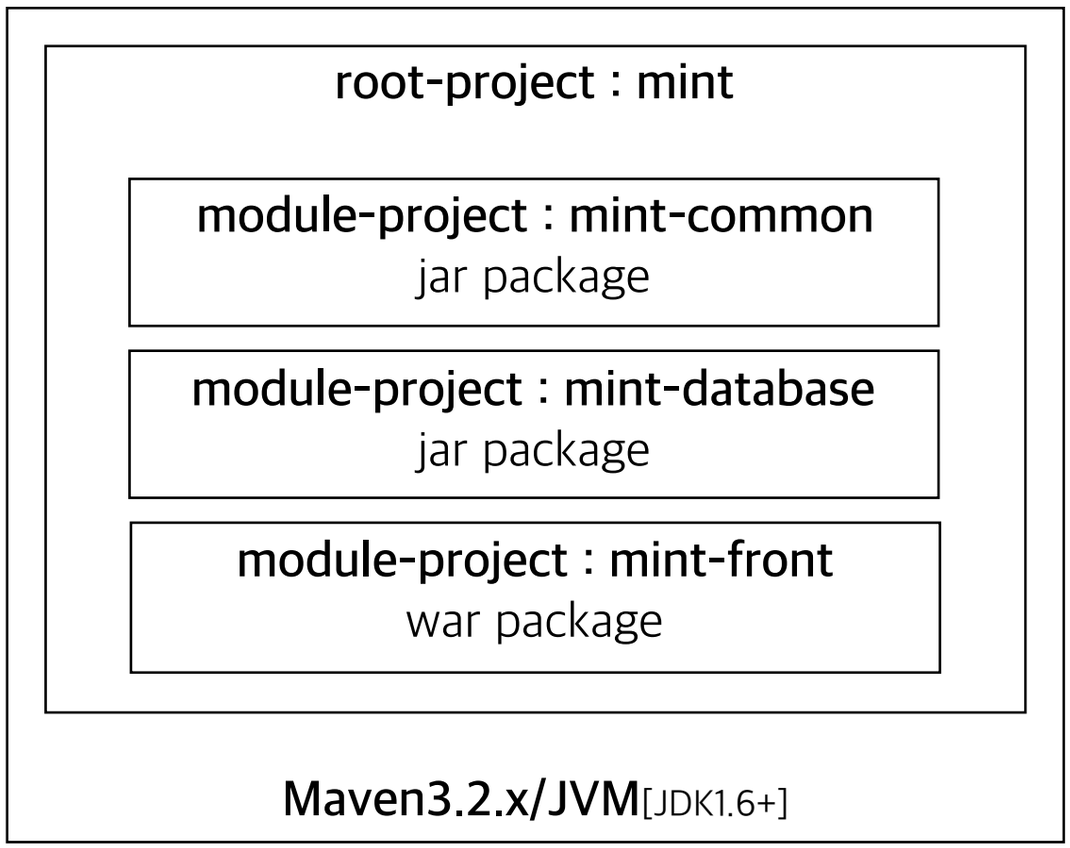
아키텍처 : 서비스 컴포넌트와 브라우저 프로그램 사이에 JSON 포맷의 Body 데이터를 HTTP 송수신 처리함.
요청 예) 요건 List 조회 요청
요청 URL : http://ips.mocomsys.com:8080/ips/an/requirements?requirementNm=부품리스트
Method : GET
응답 DATA : Requirement 오브젝트를 아이템으로 한 List

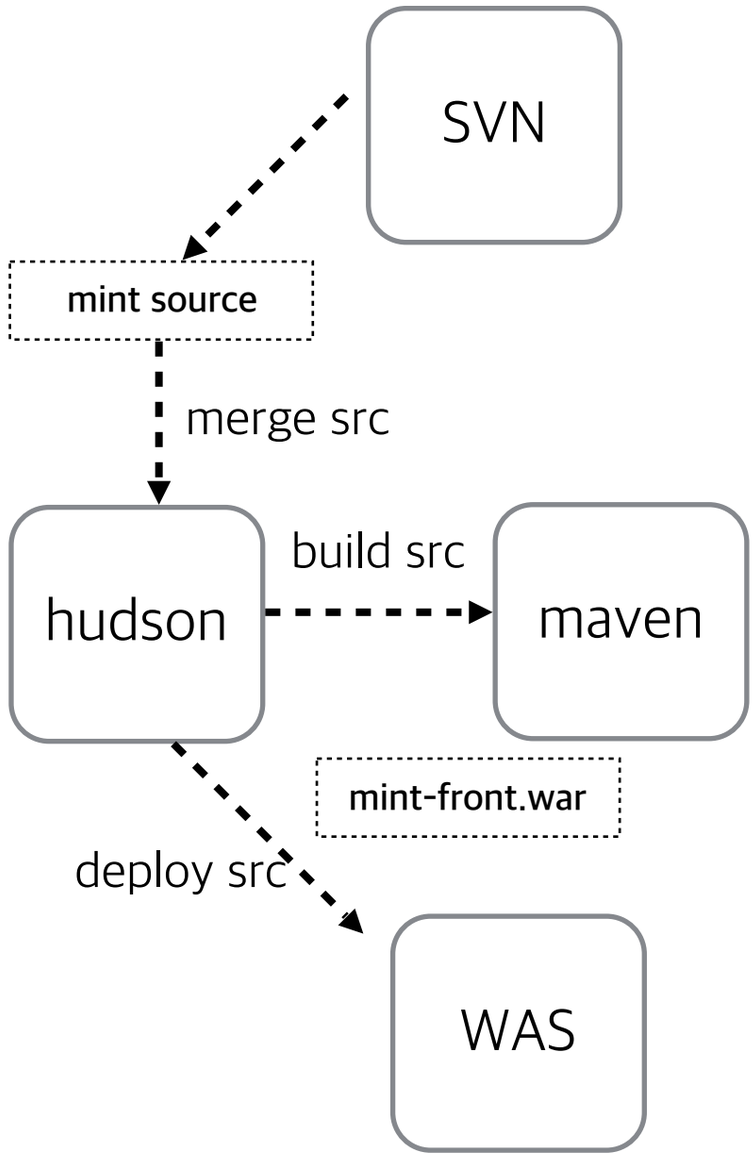


REST Service 빌드 환경

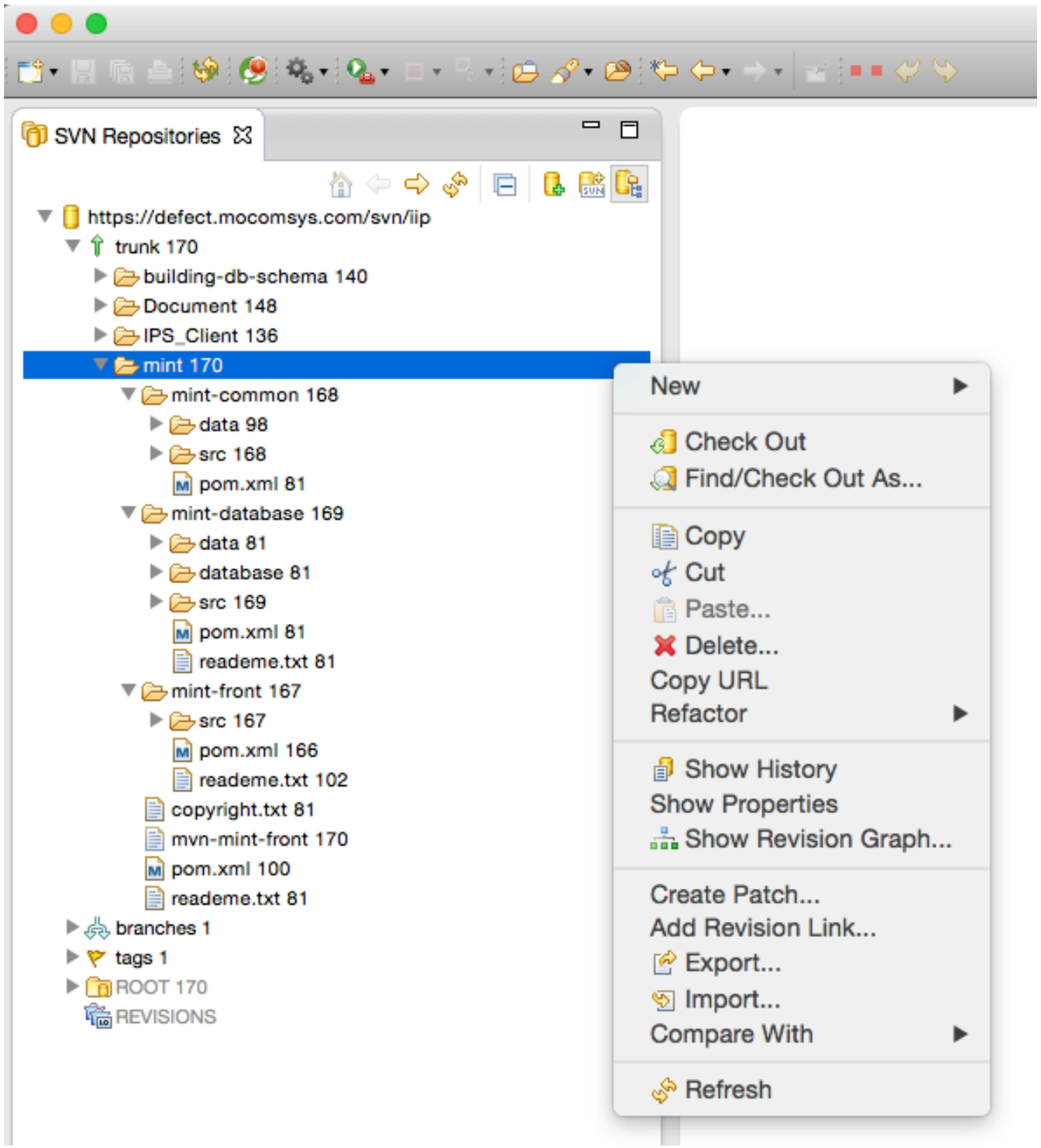
빌드환경 : 메이븐을 이용하여 빌드하며 최상위 프로젝트 mint 아래 여러개의 메이븐 모듈 프로젝트로 구성된다.

mint-common : 데이터, 유틸리티 등 공통모듈로 구성
mint-database : 데이터베이스 CRUD 서비스 모듈로 구성
mint-front : 뷰 페이지 및 REST서비스 Controller 모듈로 구성



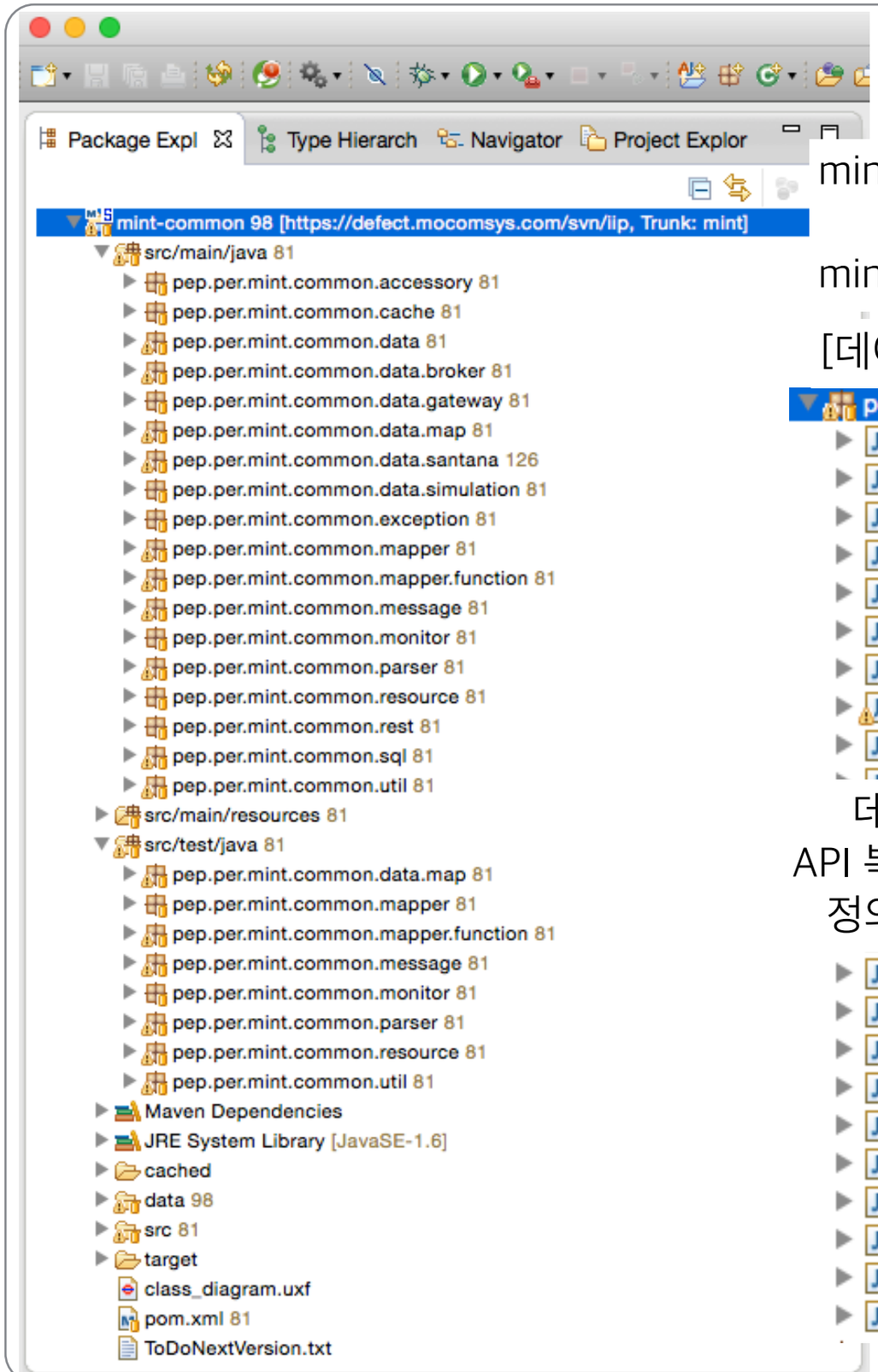


프로젝트 mint check-out



check-out 이후에 로컬에서 프로젝트를 구성한다.

프로젝트 mint-common



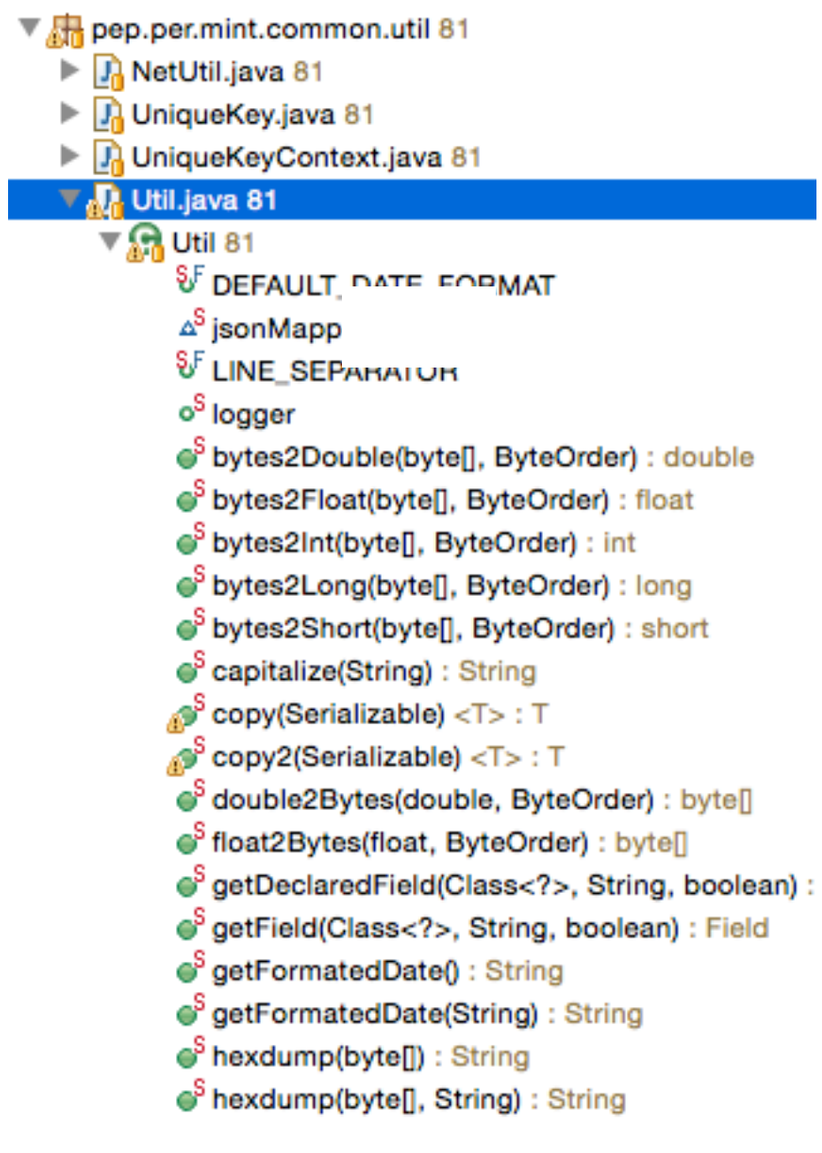
mint-common 프로젝트 모듈은 참조만하고 신규 추가는 하지 않도록 한다.

mint-common 모듈에서 주로 참조하게될 클래스 및 패키지

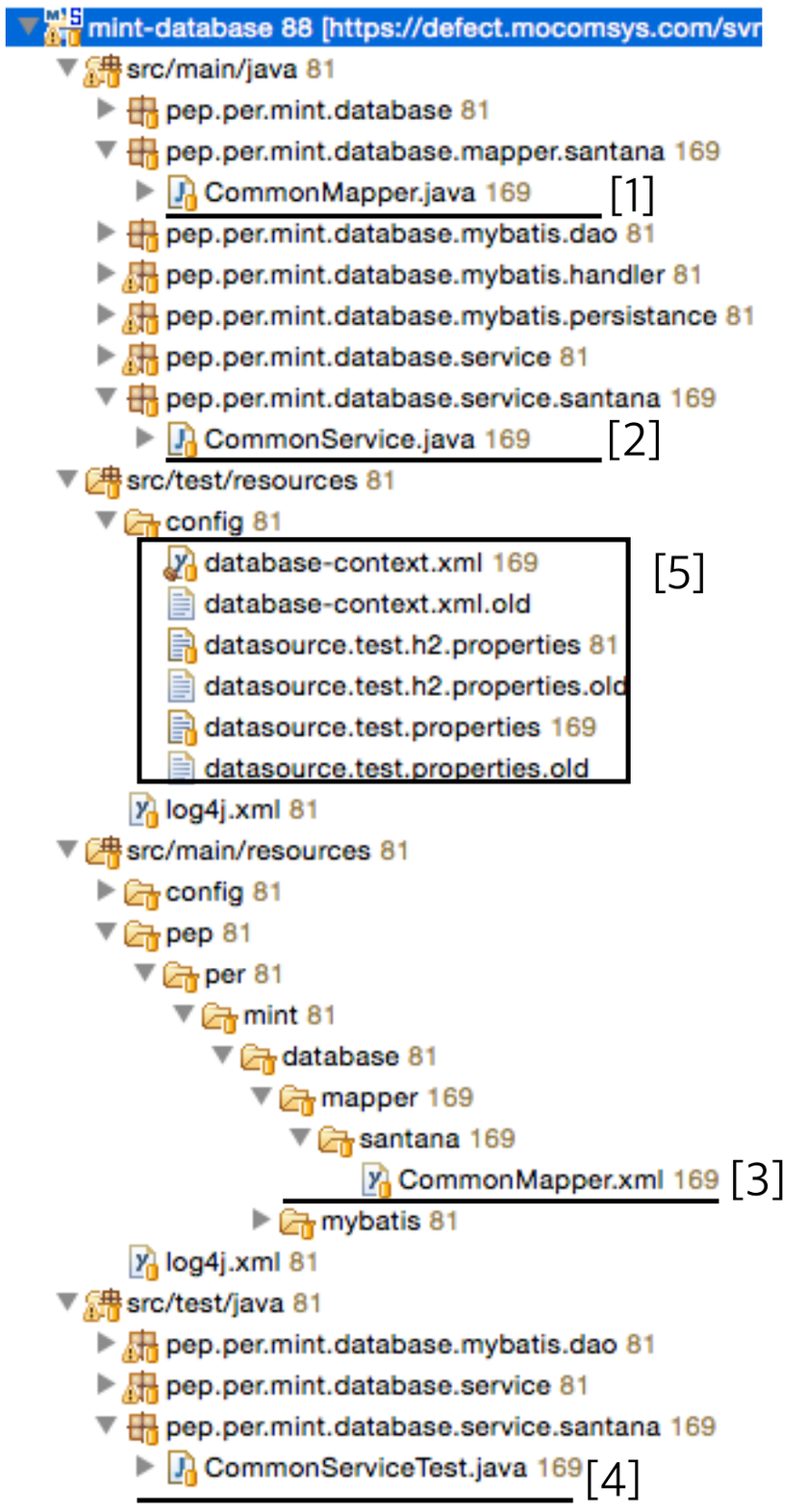
[데이터 클래스 패키지]

데이터 클래스들은
API 복합유형 정의서 상에
정의된 내용과 일치함

[Util 클래스]



프로젝트 mint-database



[mint-database 프로젝트 모듈에서 신규 서비스 추가 절차]
[1]~[3]을 신규 작성하고 [4]를 작성하여 단위 테스트 수행한다.

[1] CommonMapper.java
서비스 메소드를 정의한 Interface

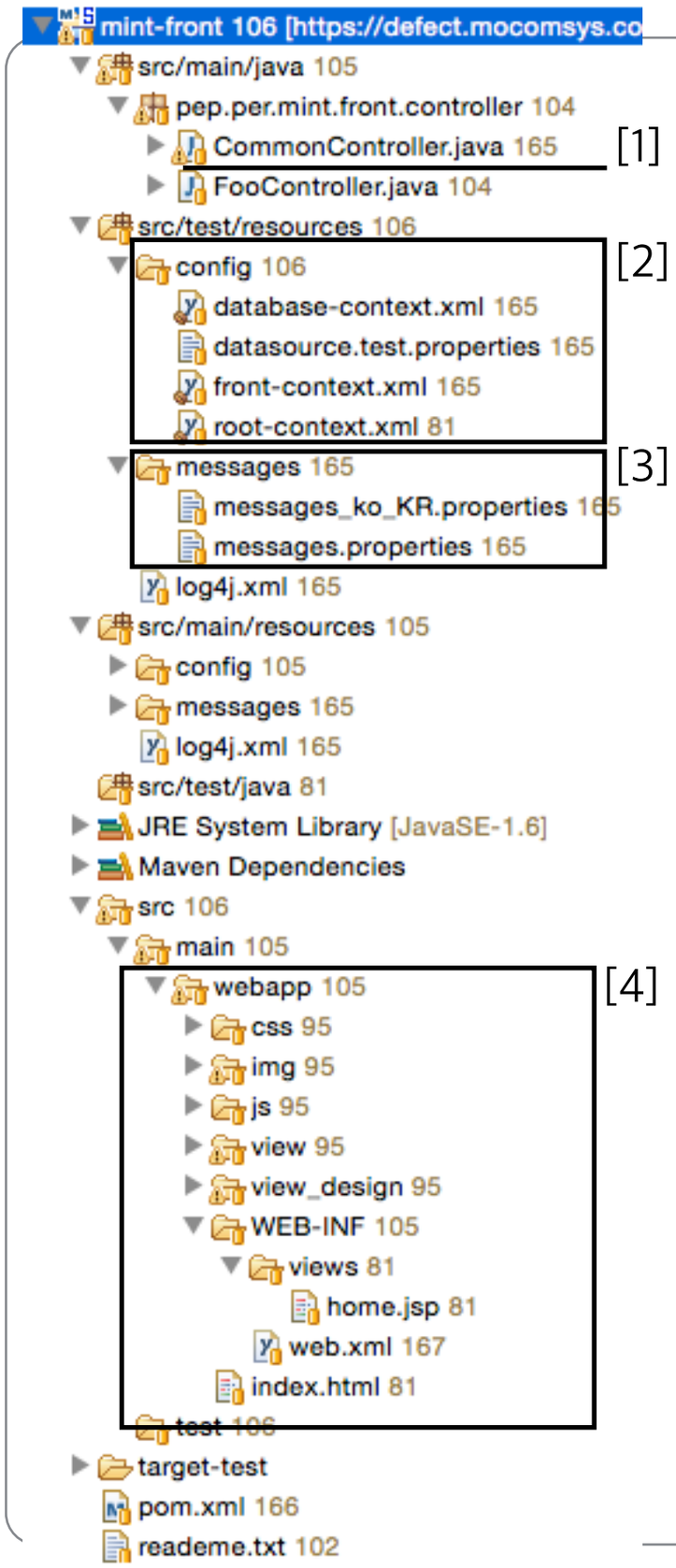
[2] CommonService.java
인터페이스에서 정의한 메소드로 실제 서비스를 구성하는
스프링 서비스 스테레오 타입 클래스

[3] CommonMapper.xml
CommonMapper.java에서 정의한 메소드에 대응하는 query를 맵핑하는
ibatis 맵퍼

[4] CommonServiceTest.java
CommonService 의 단위 테스트 클래스
main 소스와 같은 위치가 아닌 test 위치에 작성한다.(주의)
src/test/java/…

[5] config
spring context 설정 및 database 연결정보 프로퍼티

프로젝트 mint-front(1/4)



[mint-front 프로젝트 모듈 구성]

[1] CommonController.java

RESTful URI를 맵핑하고 Database 서비스를 호출하여 브라우저 요청에 대한 응답을 주는 클래스

[2] CommonService.java

spring context 설정 및 database 연결정보 프로퍼티

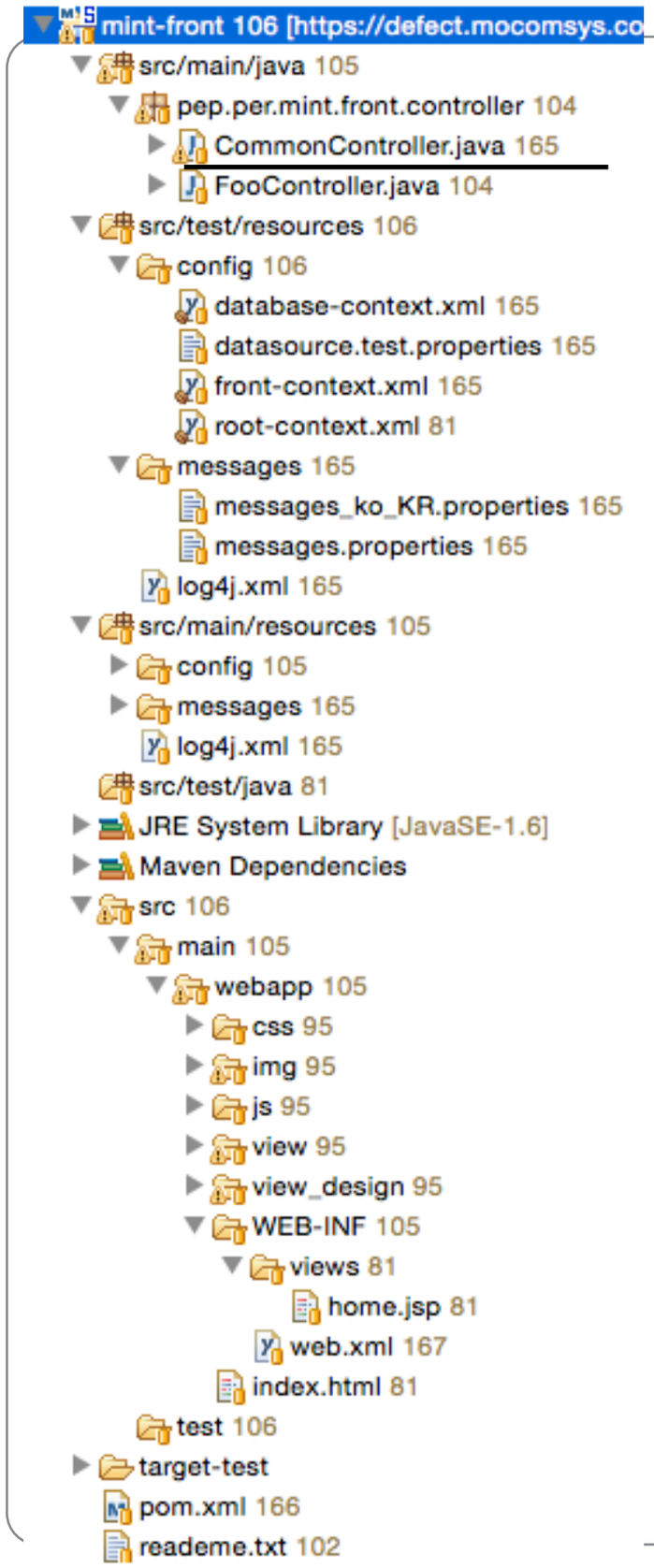
[3] messages

리소스번들을 통해 다국어 메시지를 지원하는 기능에 사용될 메시지 정의 위치
주로 Controller 내에서 비지니스 처리시 브라우저에 전달될 메시지를 다국어로 정의한다.

[4] webapp

html,jsp,css,javascript 등 뷰 리소스 위치
WEB-INF/view 는 서버에서 에러 처리나 기타 공통 처리후 리다이렉션하는 공통페이지 위치로 활용

프로젝트 mint-front(2/4)

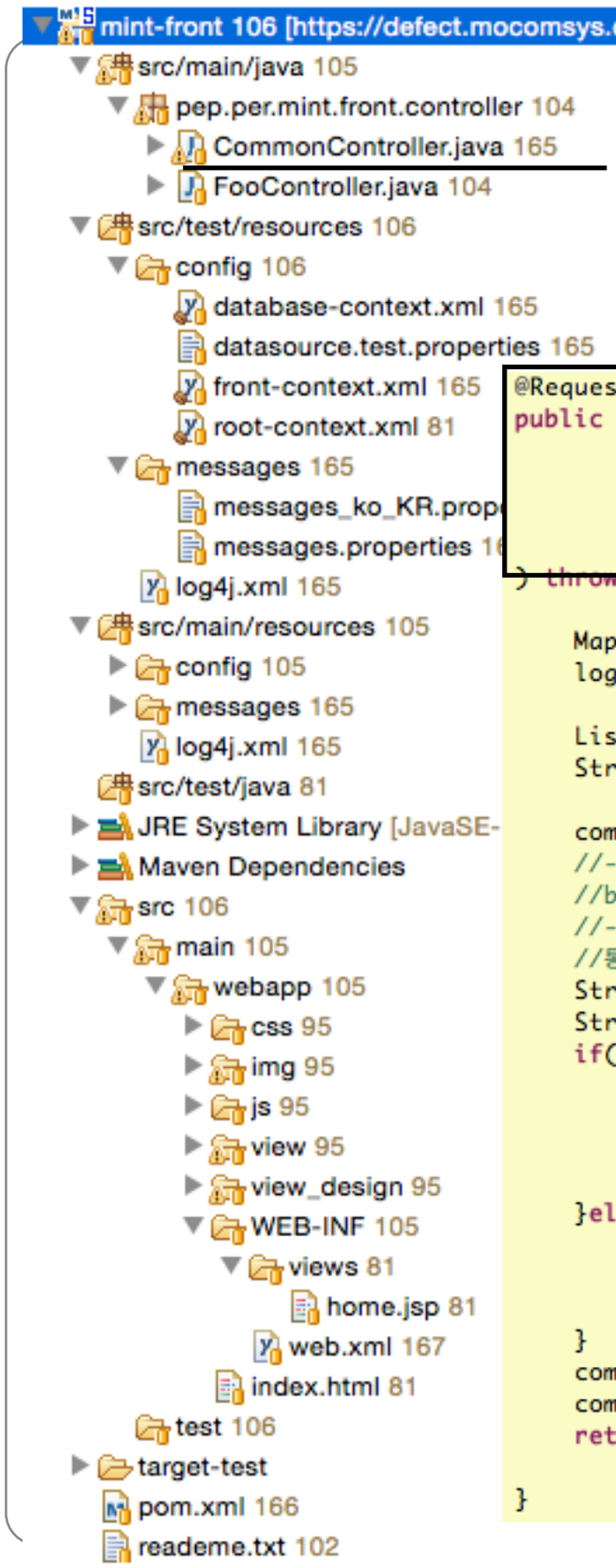


[Controller 클래스]

[1] CommonController 클래스 RequestMapping : “/co”
uri상에 유입경로가 /co로 시작되는 모든 요청에 대해 처리한다.

```java
@Controller
@RequestMapping("/co")
public class CommonController {

    private static final Logger logger = LoggerFactory.getLogger(CommonController.class);

    @Autowired
    CommonService commonService;

    @Autowired
    ReloadableResourceBundleMessageSource messageSource;

    /**
     * 공통코드 리스트 조회
     * @param level1 레벨1 코드
     * @param level2 레벨2 코드
     * @param comMessage 요청응답통신객체
     * @param locale 다국어지원을 위한 로케일
     * @return
     * @throws Exception
     */
    public @ResponseBody ComMessage<Map,List<CommonCode>> getCommonCodeList( ..

}
```

프로젝트 mint-front(3/4)



[Controller 클래스]

[1] CommonController 클래스 메소드 RequestMapping :
getCommonCodeList 메소드는 uri상에 유입 경로가
"/co/cds/{level1}/{level2}?method=GET" 로 요청되는 모든 요청에 대해 처리한다.

```
@RequestMapping(value="/cds/{level1}/{level2}", params="method=GET")
public @ResponseBody ComMessage<Map,List<CommonCode>> getCommonCodeList(
        @PathVariable String level1,
        @PathVariable String level2,
        @RequestBody ComMessage<Map,List<CommonCode>> comMessage,
        Locale locale
) throws Exception {

    Map params = comMessage.getRequestObject();
    logger.debug(Util.join("params:", Util.toJSONString(params)));

    List<CommonCode> list = commonService.getCommonCodeList(level1, level2);
    String endTime = Util.getFormatedDate("yyyyMMddHHmmssSSS");

    comMessage.setEndTime(endTime);
    //-----------------------------------------------------
    //business error process
    //-----------------------------------------------------
    //통신메시지에 처리결과 코드/메시지를 등록한다.
    String errorCd = "";
    String errorMsg = "";
    if(list == null || list.size() == 0){//결과가 없을 경우 비지니스 예외 처리
        logger.debug(Util.join("default locale:",locale.toString(), ",", locale.getLanguage(),",",locale.getCountry()));

        errorCd = messageSource.getMessage("error.cd.result.none", null, locale);
        errorMsg= messageSource.getMessage("error.msg.result.none", null, locale);
    }else{//성공 처리결과
        errorCd = messageSource.getMessage("error.cd.ok", null, locale);
        errorMsg= messageSource.getMessage("error.msg.ok", null, locale);
        comMessage.setResponseObject(list);
    }
    comMessage.setErrorCd(errorCd);
    comMessage.setErrorMsg(errorMsg);
    return comMessage;

}
```

프로젝트 mint-front(4/4)

[Controller 클래스 테스트 메소드 작성]

database 서비스 개발전에 front로부터의 요청을 제대로 받아들이는지 확인하고 테스트 데이터를 응답해 주는 단위 테스트 함수 작성법을 설명한다.
테스트 요청을 보낼 때는 본래 URI 상에 파라메터로 “isTest=true”를 추가하여 호출한다.

```
107⊕ public @ResponseBody ComMessage<Map,List<CommonCode>> getCommonCodeList( ..
143
144
146⊕  * <pre>..
```

실제 Database 서비스를 이용하여 공통코드를 조회하는 메소드

testGetCommonCodeList 메소드 작성부

```
157⊖ @RequestMapping(value="/cds/{level1}/{level2}", params={"method=GET", "isTest=true"})
158  public @ResponseBody ComMessage<Map,List<CommonCode>> testGetCommonCodeList(
159          @PathVariable String level1,
160          @PathVariable String level2,
161          @RequestBody ComMessage<Map,List<CommonCode>> comMessage,
162          Locale locale
163  ) throws Exception {
164
165      String testFilePath = servletContext.getRealPath("/WEB-INF/test-data/co/REST-R01-CO-02-00-005.json");
166      logger.debug(Util.join("testFilePath:", testFilePath));
167      List<CommonCode> list = (List<CommonCode>) Util.readObjectFromJson(new File(testFilePath), List.class, "UTF-8");
168      String endTime = Util.getFormatedDate("yyyyMMddHHmmssSSS");
169      comMessage.setEndTime(endTime);
170      //-----------------------------------------------------------
171      //business error process
172      //-----------------------------------------------------------
173      //통신메시지에 처리결과 코드/메시지를 등록한다.
174      String errorCd = "";
175      String errorMsg = "";
176      if(list == null || list.size() == 0){//결과가 없을 경우 비지니스 예외 처리
177          logger.debug(Util.join("default locale:",locale.toString(), ",", locale.getLanguage(),",",locale.getCountry()));
178          errorCd = messageSource.getMessage("error.cd.result.none", null, locale);
179          errorMsg= messageSource.getMessage("error.msg.result.none", null, locale);
180      }else{//성공 처리결과
181          errorCd = messageSource.getMessage("error.cd.ok", null, locale);
182          errorMsg= messageSource.getMessage("error.msg.ok", null, locale);
183          comMessage.setResponseObject(list);
184      }
185      comMessage.setErrorCd(errorCd);
186      comMessage.setErrorMsg(errorMsg);
187      return comMessage;
188  }
```

Database 서비스 개발전 getCommonCodeLIst 메소드 요청에 대한 테스트 응답을 돌려주는 Controller 테스트 메소드 작성부분

공통코드 테스트 데이터 JSON 포멧 파일

JSON 포멧 파일을 읽어들여 응답객체로 바인딩한다.

REST Service & Mapper 클래스 정의

[Service 정의]

* 클래스 네이밍은 카멜표기법을 따른다.

분류	패키지	클래스	설명
[AN]	pep.per.mint.database.service.an	RequirementService	요건을 중심으로한 서비스를 제공한다.
		InterfaceService	인터페이스를 중심으로한 서비스를 제공한다.
		RuleService	룰,룰셋을 중심으로한 서비스를 제공한다.
[IM]	pep.per.mint.database.service.im	InfraService	시스템,서버,업무를 중심으로한 서비스를 제공한다.
		AppService	어플리케이션(APP)를 중심으로한 서비스를 제공한다.
		SimulationService	시뮬레이션 테스트를 중심으로한 서비스를 제공한다.
[OP]	pep.per.mint.database.service.op	MonitorService	트레킹, 거래로그, 모니터링을 중심으로한 서비스를 제공한다.
		DashboardService	데시보드를 중심으로한 서비스를 제공한다.
		ProblemService	장애관리를 중심으로한 서비스를 제공한다.
[SU]	pep.per.mint.database.service.su	EnvironmentService	환경설정을 중심으로한 서비스를 제공한다.
		CollaboService	협업을 중심으로한 서비스를 제공한다.
		SummaryService	집계/현황 중심으로한 서비스를 제공한다.
[CO]	pep.per.mint.database.service.co	CommonService	공통코드,시스템,업무,서버 코드 조회등 공통을 중심으로한 서비스를 제공한다.

REST Service & Mapper 클래스 정의

[Mapper 정의]

* 클래스 네이밍은 카멜표기법을 따른다.

분류	패키지	클래스	설명
[AN]	pep.per.mint.database.mapper.an	RequirementMapper	요건을 중심으로한 서비스를 제공한다.
		InterfaceMapper	인터페이스를 중심으로한 서비스를 제공한다.
		RuleMapper	룰,룰셋을 중심으로한 서비스를 제공한다.
[IM]	pep.per.mint.database.mapper.im	InfraMapper	시스템,서버,업무를 중심으로한 서비스를 제공한다.
		AppMapper	어플리케이션(APP)를 중심으로한 서비스를 제공한다.
		SimulationMapper	시뮬레이션 테스트를 중심으로한 서비스를 제공한다.
[OP]	pep.per.mint.database.mapper.op	MonitorMapper	트레킹, 거래로그, 모니터링을 중심으로한 서비스를 제공한다.
		DashboardMapper	데시보드를 중심으로한 서비스를 제공한다.
		ProblemMapper	장애관리를 중심으로한 서비스를 제공한다.
[SU]	pep.per.mint.database.mapper.su	EnvironmentMapper	환경설정을 중심으로한 서비스를 제공한다.
		CollaboMapper	협업을 중심으로한 서비스를 제공한다.
		SummaryMapper	집계/현황 중심으로한 서비스를 제공한다.
[CO]	pep.per.mint.database.mapper.co	CommonMapper	공통코드,시스템,업무,서버 코드 조회등 공통을 중심으로한 서비스를 제공한다.

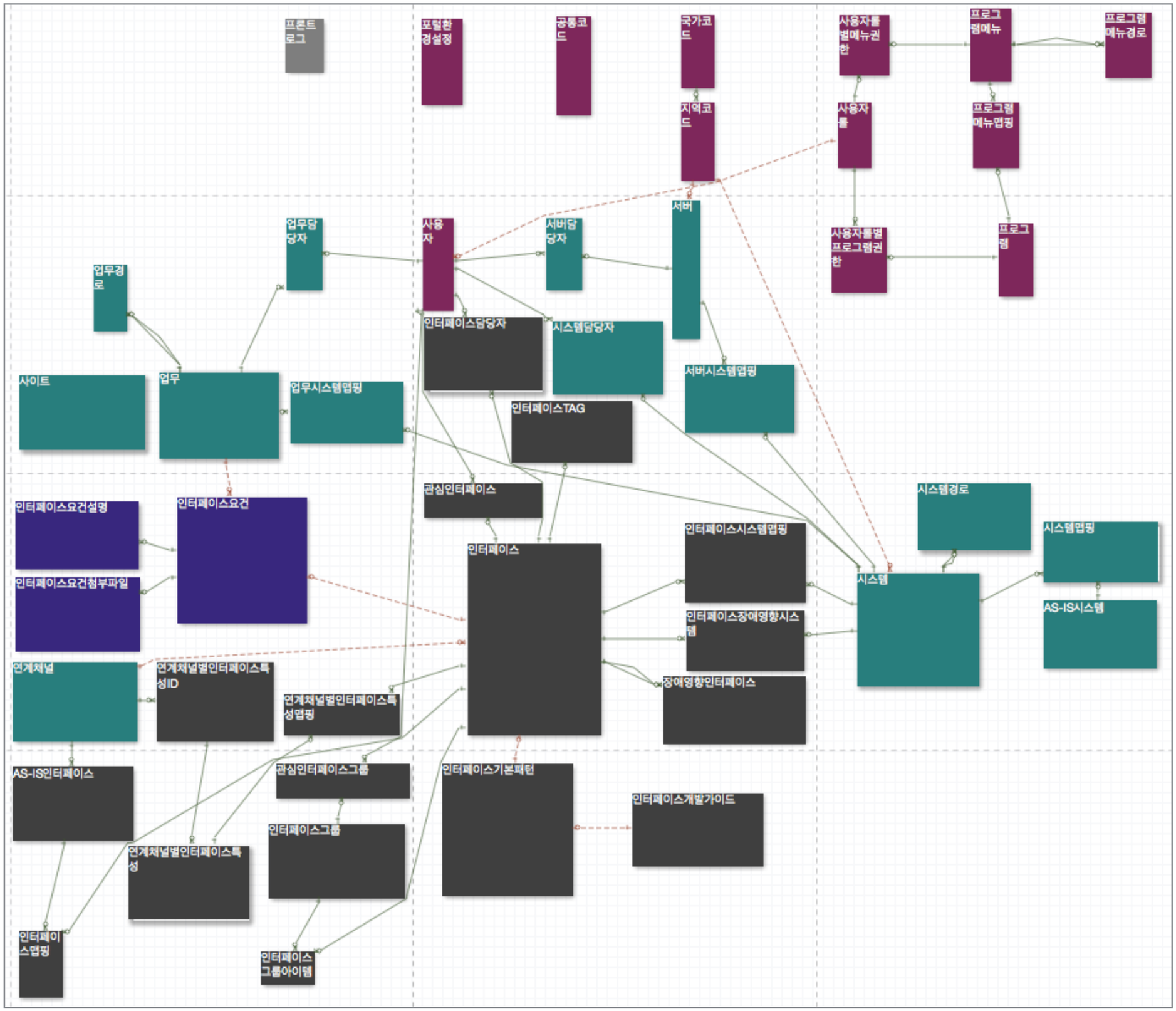
REST Front Controller 정의

[Controller 정의]

* 클래스 네이밍은 카멜표기법을 따른다.

분류	패키지	클래스	설명
[AN]	pep.per.mint.front.controller.an	RequirementController	요건을 중심으로한 서비스를 제공한다.
		InterfaceController	인터페이스를 중심으로한 서비스를 제공한다.
		RuleController	룰,룰셋을 중심으로한 서비스를 제공한다.
[IM]	pep.per.mint.front.controller.im	InfraController	시스템,서버,업무를 중심으로한 서비스를 제공한다.
		AppController	어플리케이션(APP)를 중심으로한 서비스를 제공한다.
		SimulationController	시뮬레이션 테스트를 중심으로한 서비스를 제공한다.
[OP]	pep.per.mint.front.controller.op	MonitorController	트레킹, 거래로그, 모니터링을 중심으로한 서비스를 제공한다.
		DashboardController	데시보드를 중심으로한 서비스를 제공한다.
		ProblemManageController	장애관리를 중심으로한 서비스를 제공한다.
[SU]	pep.per.mint.front.controller.su	EnvironmentController	환경설정을 중심으로한 서비스를 제공한다.
		CollaboController	협업을 중심으로한 서비스를 제공한다.
		SummaryController	집계/현황 중심으로한 서비스를 제공한다.
[CO]	pep.per.mint.front.controller.co	CommonController	공통코드,시스템,업무,서버 코드 조회등 공통을 중심으로한 서비스를 제공한다.

DATABASE SCHEMA 설명(1/3)



DATABASE SCHEMA 설명(2/3)

PREFIX	대분류	중분류	상세항목	PHYSICAL	LOGICAL	COMMENTS	삭제	
T	AN	01	01	TAN0101	인터페이스요건	인터페이스개발요건을 관리하기위한 Entity		
T	AN	01	02	TAN0102	인터페이스요건설명	인터페이스개발요건의 부가설명 또는 담당자끼리의 문답 내용을 관리하기위한 Entity		
T	AN	01	03	TAN0103	인터페이스요건첨부파일	인터페이스개발요건의 첨부파일을 관리하기위한 Entity		
T	AN	01	03	TAN0103	사이트별요건특성ID	사이트별로 인터페이스요건의 틀린 특성ID들을 관리하기 위한 Entity	O	쓰이지 않을 것으로
T	AN	01	04	TAN0104	사이트별요건특성	사이트별로 인터페이스요건의 틀린 특성값들을 관리하기 위한 Entity	O	쓰이지 않을 것으로
T	AN	02	01	TAN0201	인터페이스	인터페이스요건으로부터 식별되어 도출된 인터페이스를 관리하기위한 Entity		
T	AN	02	02	TAN0202	요건인터페이스맵핑	인터페이스의 장애 영향도를 표현하는 속성을 관리하는 Entity		
T	AN	02	03	TAN0203	인터페이스장애영향도특성	인터페이스의 장애 영향도를 표현하는 속성을 관리하는 Entity	O	정규화 (인터페이스
T	AN	02	04	TAN0204	연계채널별인터페이스특성ID	연계시스템별 인터페이스요건의 틀린 특성ID들을 관리하기 위한 Entity		
T	AN	02	05	TAN0205	연계채널별인터페이스특성	연계시스템별로 인터페이스요건의 틀린 특성값들을 관리하기 위한 Entity		
T	AN	02	06	TAN0206	인터페이스개발패턴	사용자가 선택 또는 정의한 인터페이스개발 패턴을 관리하는 Entity	O	정규화 (인터페이스
T	AN	02	07	TAN0207	인터페이스기본패턴	표준 인터페이스개발 패턴을 관리하는 Entity		
T	AN	02	08	TAN0208	인터페이스개발가이드	표준 인터페이스개발 패턴의 개발 가이드를 관리하는 Entity		
T	AN	02	09	TAN0209	장애영향인터페이스	장애발생시 서로 영향을 주는 인터페이스를 관리하는 Entity		
T	AN	02	10	TAN0210	인터페이스맵핑	표준인터페이스와 기존(AS-IS)인터페이스의 맵핑정보를 관리하는 Entity 맵핑키는 인터페이스ID 와 기존 AS-IS 인터페이스ID 이다.		
T	AN	02	11	TAN0211	AS-IS인터페이스	인터페이스 표준화 이전에 존재하던 시스템에서 식별된 기존 인터페이스 정보를 관리하기위한 Entity 주로 인터페이스맵핑정보로 활용하기위한 목적으로 쓰인다.		
T	AN	02	12	TAN0212	인터페이스장애영향시스템	인터페이스 장애발생시 영향받는 시스템을 관리하는 Entity		삭제여부 판단필요 장애영향인터페이스
T	AN	02	13	TAN0213	인터페이스시스템맵핑	인터페이스와 인터페이스가 연계할 시스템들을 관리할 Entity		
T	AN	02	14	TAN0214	관심인터페이스	사용자가 관심갖는 인터페이스를 관리하는 Entity		
T	AN	02	15	TAN0215	인터페이스그룹	특정목적의 인터페이스를 그룹핑하여 관리할수있도록 하는 Entity 아래와 같은 확장기능을 염두해두고 설계한다. 1) 업무 시스템 등과 별도의 관련 인터페이스 그룹핑 기능 제공 2) 워크플로우 단위 인터페이스 관리가 필요할 경우 활요 3) 스케쥴링 데이터와 연계하여 잡스케쥴링 기능 제공 4) 인터페이스그룹 처리상태 표현		
T	AN	02	16	TAN0216	인터페이스그룹아이템	인터페이스그룹을 구성하는 인터페이스 리스트를 관리하는 Entity		
T	AN	02	17	TAN0217	관심인터페이스그룹	인터페이스그룹을 구성하는 인터페이스 리스트를 관리하는 Entity		
T	AN	02	18	TAN0218	인터페이스업무맵핑	사용자가 관심갖는 인터페이스그룹을 관리하는 Entity		
T	AN	02	19	TAN0219	인터페이스담당자	인터페이스 담당자들을 관리하는 Entity		
T	AN	02	20	TAN0220	연계채널별인터페이스특성맵핑	연계채널별로관리되는 인터페이스특성을 인터페이스와 맵핑 관리하는 Entity		
T	AN	02	21	TAN0221	인터페이스TAG	인터페이스 검색시 사용될 TAG 관리 Entity		

DATABASE SCHEMA 설명(3/3)

ERD 세부 설명

REST Service/Controller 개발 절차

REST API 설계서 식별

REST API 설계서 상 개발 내역 확인

Database 서비스 개발

1) FooMapper 인터페이스 작성
2) FooService 클래스 작성
3) FooMapper.xml 맵핑 작성
4) FooServiceTest 테스트 클래스 작성 및 테스트
5) 빌드

Controller 개발

1) FooController 작성
2) 빌드 & 로컬 WAS 배포
3) 테스트 데이터 작성 & 호출 테스트

배포

1) 개발 소스 commit
2) 개발/배포 관리자에게 노티
3) 개발기 배포
4) 2)~3)은 허드슨을 이용한 자동화 처리 무방한가?

REST Service/Controller 작성 실습

? 알아서

2.서비스 정의

- API 코드 네이밍 : REST-[CRUDPS][SEQ]-[리소스]
  - 구분자 : 하이픈 (-)
  - prefix : REST
  - CRUDPS : C: Create, R: Retrieve, U: Update, D: Delete , S: Service, P: Print
  - SEQ : 두자리 패딩 순번
  - 리소스 : Menu-1, Menu-2, Menu-3, Screen 으로 구성된 리소스로 식별가능한 물리적 또는 논리적 화면ID중 뒤 3자리를 제외한 값을 사용하며 리소스로 표현함

※ 단계코드 : 분석(Analysis), 설계(Design), 구현(Implementation), 테스트(Test)

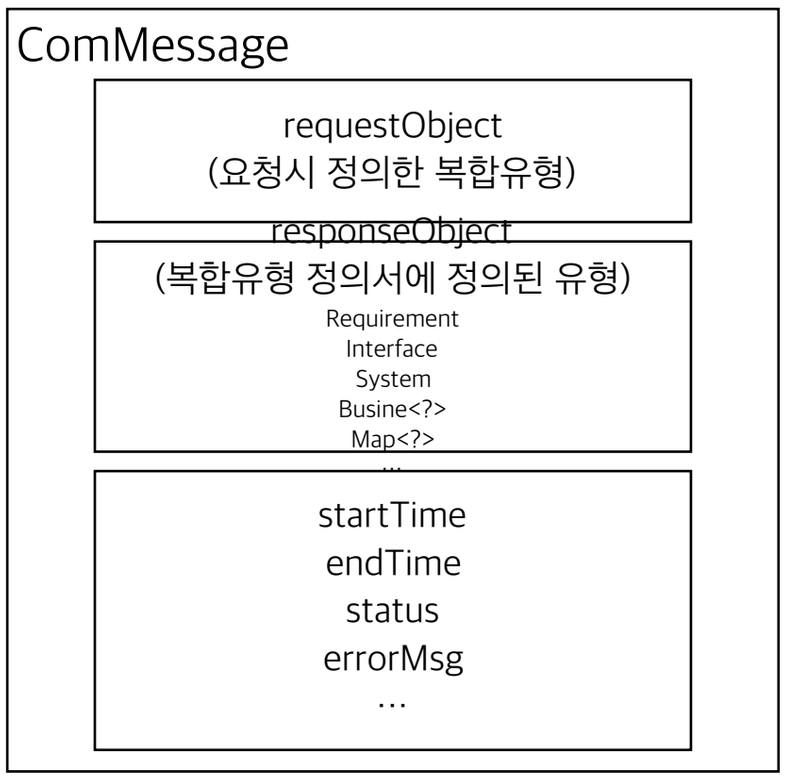
22	개발	IM	인프라	01	시스템(그룹)	01	시스템(그룹) 등록/상세	002	IM-01-01-002	시스템(그룹) 등록	REST-C01-IM-01-01
23	개발	IM	인프라	01	시스템(그룹)	01	시스템(그룹) 등록/상세	002	IM-01-01-002	시스템(그룹) 수정	REST-U01-IM-01-01
24	개발	IM	인프라	01	시스템(그룹)	01	시스템(그룹) 등록/상세	002	IM-01-01-002	시스템(그룹) 삭제	REST-D01-IM-01-01
25	개발	IM	인프라	01	서버(그룹)	02	서버(그룹) 조회	001	IM-01-02-001	서버 리스트 조회	REST-R01-IM-01-02
26	개발	IM	인프라	01	서버(그룹)	02	서버(그룹) 조회	001	IM-01-02-001	서버 트리 조회	REST-R03-IM-01-02
27	개발	IM	인프라	01	서버(그룹)	02	서버(그룹) 등록/상세	002	IM-01-02-002	서버 상세 조회	REST-R02-IM-01-02
28	개발	IM	인프라	01	서버(그룹)	02	서버(그룹) 등록/상세	002	IM-01-02-002	서버 등록	REST-C01-IM-01-02
29	개발	IM	인프라	01	서버(그룹)	02	서버(그룹) 등록/상세	002	IM-01-02-002	서버 수정	REST-U01-IM-01-02
30	개발	IM	인프라	01	서버(그룹)	02	서버(그룹) 등록/상세	002	IM-01-02-002	서버 삭제	REST-D01-IM-01-02
31	개발	IM	인프라	01	업무(그룹)	03	업무(그룹) 조회	001	IM-01-03-001	업무(그룹) 리스트 조회	REST-R01-IM-01-02
32	개발	IM	인프라	01	업무(그룹)	03	업무(그룹) 조회	001	IM-01-03-001	업무 트리 조회	REST-R03-IM-01-02
33	개발	IM	인프라	01	업무(그룹)	03	업무(그룹) 등록/상세	002	IM-01-03-002	업무(그룹) 상세 조회	REST-R02-IM-01-02
34	개발	IM	인프라	01	업무(그룹)	03	업무(그룹) 등록/상세	002	IM-01-03-002	업무(그룹) 등록	REST-C01-IM-01-02
35	개발	IM	인프라	01	업무(그룹)	03	업무(그룹) 등록/상세	002	IM-01-03-002	업무(그룹) 수정	REST-U01-IM-01-02
36	개발	IM	인프라	01	업무(그룹)	03	업무(그룹) 등록/상세	002	IM-01-03-002	업무(그룹) 삭제	REST-D01-IM-01-02
37	개발	IM	런타임	02							
38	개발	IM	시뮬레이터	03							
39	운영	OP	모니터링	01	Interface Tracking	01	Interface Tracking	00			
40	운영	OP	모니터링	01	Interface Tracking	01	Interface Tracking-Detail	00			
41	운영	OP	모니터링	01	자원 모니터링	02					
42	운영	OP	대시보드	02	N/A	00	메인대시보드	001	OP-02-00-001		
43	운영	OP	장애관리	03	N/A	00	인터페이스 영향도	001	OP-03-00-001		
44	운영	OP	장애관리	03	N/A	00	시스템 영향도	002	OP-03-00-002		
45	운영	OP	장애관리	03	N/A	00	장애대장	003			
46	운영	OP	장애관리	03	N/A	00	장애 히스토리 관리	004			
47	지원	SU	포탈환경설정	01	사용자 관리	01	사용자 조회	001			
48	지원	SU	포탈환경설정	01	사용자 관리	01	사용자 등록/상세	002			
49	지원	SU	포탈환경설정	01	권한 관리	02	권한 조회	001			
50	지원	SU	포탈환경설정	01	권한 관리	02	권한 등록/상세	002			
51	지원	SU	포탈환경설정	01	공통 코드	03	공통 코드 조회	001			
52	지원	SU	포탈환경설정	01	공통 코드	03	공통 코드 등록/상세	002			
53	지원	SU	포탈환경설정	01	메뉴 관리	04	메뉴 조회	001			
54	지원	SU	포탈환경설정	01	메뉴 관리	04	메뉴 등록/상세	002			
55	지원	SU	포탈환경설정	01	초기화면 설정	05	초기화면 설정 조회	001			
56	지원	SU	포탈환경설정	01	초기화면 설정	05	초기화면 설정 등록	002			
57	지원	SU	협업	02	승인/결재 연동	01					
58	지원	SU	협업	02	SSO/사용자 연동	02					
59	지원	SU	협업	02	비지니스/메타 연동	03					
60	지원	SU	협업	02	게시/알림 연동	04					
61	지원	SU	통계/KPI	03	N/A	00	표준화 준수율				
62	지원	SU	통계/KPI	03	N/A	00	인터페이스 재 사용률				
63	지원	SU	통계/KPI	03	진척률 통계	01					

REST API LIST 문서 참고

통신객체 정의

표준 통신객체를 정의하여 서버 측 에러 외에 업무적 에러 발생 내용을 Front 측에 전달하여 화면 처리시 참고 할 수 있도록 한다.
이하 설계되는 모든 API는 ComMessage 내에 comObject 필드에 요청/응답 객체를 담아서 Request/Response 되는 것으로 규정한다.

ComMessage 구조



ComMessage 복합유형 정의

필드ID	필드명	유형	설명
objectType	오브젝트유형	String	고정값 : “ComMessage”
requestObject	요청 객체	Complex	
responseObject	응답 객체	Complex	
startTime	처리시작시간	String	요청시간 : 형식 YYYYMMDDHH24MISSsss
endTime	처리종료시간	String	응답시간 : 형식 YYYYMMDDHH24MISSsss
errorCd	처리결과	String	응답에서 채워줌
errorMsg	에러메시지	String	응답에서 채워줌
userId	요청자ID	String	요청 USER ID
appId	요청화면ID	String	화면 APP ID
checkSession	세션체크구분	boolean	false 면 서비스에서 세션체크 skip (단위테스트용)

ComMessage 에러코드[errorCd] 필드

0 : 성공
0 이외의 값은 에러로 판단한다. 필요하다면 에러코드 별도 정의

HTTP Method

HTTP Method 헤더값 대신 URLString 에 key/value 파라메터값을 사용한다. (?method=GET)
메소드 값 : Create : **POST,** Retrieve : **GET,** Update : **PUT,** Delete : **DELETE**
오퍼레이션 확장 : PRINT, EXPORT-EXCEL, EXPORT-PDF 등 대문자 및 “-“ 를 사용하여 CRUD 외의 오퍼레이션 명을 확장 정의하여 사용한다.
HTTP 헤더 METHOD를 이용하지 않는 이유 : REST 양식의 오퍼레이션 식별 방식을 이용하려 하였으나, 조회 조건을 URI상에 표현하는데 길이제약 및 한글처리의 번거로움 등 현실적인 한계가 있어 모든 오퍼레이션은 **HTTP.Method = POST 방식으로 통일하기로 하며** ComMessage의 method필드로 서버에서 수행할 오퍼레이션 값을 전달하기로 규정한다.

통신객체 정의

API 호출 예

http://127.0.0.1:8080/mint/co/cds/IM/040?method=GET

Content-Type=application/json;charset=UTF-8
HTTP.Method=POST
*주의사항 : HTTP.Method 는 항상 POST 로 고정하여 호출한다.(REST 리소스 오퍼레이션은 URI 상에 파라미터 method를 이용하기로 함)

ComMessage 요청

```
{
  "objectType": "ComMessage",
  "requestObject":null,
  "startTime": "20150701120001001",
  "endTime": null,
  "errorCd":"0",
  "errorMsg":"",
  "userId": "whoana",
  "appId": "AN-01-00-001"
}
```

*주의사항 : 요청 메시지의 requestObject 에 할당되는 복합유형에 objectType 필드 값이 있어야만 서버측에서 RequestBody를 자바 객체로 변환하는데 문제없는 지 확인 필요함.
없어도 변환상에 문제가 없다면 넣을 필요는 없다.

ComMessage 응답

```
{
  "objectType": "ComMessage",
  "id": null,
  "requestObject": null,
  "responseObject": [
   {
    "objectType": "CommonCode",
    "id": null,
    "level1": "IM",
    "level2": "04",
    "cd": "0",
    "level1Nm": "개발",
    "level2Nm": "리소스",
    "nm": "Database",
    "nm2": "DB",
    "comments": "개발-리소스-Database",
    "delYn": "N",
    "regDate": "20150703120050",
    "regId": "iip",
    "modDate": "20150703120050",
    "modId": "iip"
   },
   {
    "objectType": "CommonCode"
```

요건 리스트 조회

API ID	REST-R01-AN-01-00	Controller Class	pep.per.mint.front.controller.an.RequirementController
API Name	요건 리스트 조회	Controller Method	getRequirementList
URI	/an/requirements	Service Class	per.per.mint.database.service.an.RequirementService
method	GET	Service Method	getRequirementList
appId	AN-01-00-001	Mapper Class	per.per.mint.database.mapper.an.RequirementMapper

Request Object

필드ID	필드명	유형	NOT NULL
channelId	연계채널ID	String	
status	상태	String	
businessId	업무	String	
requirementNm	요건명	String	
requirementId	요건ID	String	
interfaceNm	인터페이스명	String	
integrationId	통합인터페이스ID	String	
systemNm	시스템명	String	
service	서비스	String	
resourceNm	시스템 리소스명	String	
tag	태그	String	
isRelUser	담당자구분	String	O

Response Object

List<Requirement>

설명

1요건 리스트를 조회한다.
2.Request Object 는 서버측에서 조회 조건으로 사용된다.
3.조회 조건은 값이 업을 경우 필드ID는 전달하지 않도록 한다.
4.조회결과로 리턴될 복합유형은 리스트 화면에 보여줄 최소한의 필드 데이터로와 상세조회시 반드시 필요한 키값 으로 구성하여 대량 데이터 조회시 부하문제가 없도록 한다.

요건삭제

API ID	REST-D01-AN-01-00	Controller Class	pep.per.mint.front.controller.an.RequirementController
API Name	요건 삭제	Controller Method	deleteRequirement
URI	/an/requirements/{requirementId}	Service Class	per.per.mint.database.service.an.RequirementService
method	DELETE	Service Method	deleteRequirement
appId	AN-01-00-002	Mapper Class	per.per.mint.database.mapper.an.RequirementMapper
Request Object			
Response Object	Requirement		
설명	1.요건을 삭제 한다. 2.삭제시 연관된 엔터티도 삭제하도록 한다. 3.인터페이스의 경우 레퍼런스 되고 있는 상태라면 그냥 둔다. 4.요건 리스트를 삭제할 경우는 요건 삭제 API를 여러번 반복 호출한다.		

요건등록

API ID	REST-C01-AN-01-00	Controller Class	pep.per.mint.front.controller.an.RequirementController
API Name	요건 등록	Controller Method	createRequirement
URI	/an/requirements	Service Class	per.per.mint.database.service.an.RequirementService
method	POST	Service Method	createRequirement
appId	AN-01-00-002	Mapper Class	per.per.mint.database.mapper.an.RequirementMapper
Request Object	Requirement		
Response Object	Requirement		
설명	1.요건을 등록 한다. 2.요청시 요건 ID 값은 null로 요청하여 서버에서 자동 채번 됨		

요건변경

API ID	REST-U01-AN-01-00	Controller Class	pep.per.mint.front.controller.an.RequirementController
API Name	요건 변경	Controller Method	updateRequirement
URI	/an/requirements/{requirementId}	Service Class	per.per.mint.database.service.an.RequirementService
method	PUT	Service Method	updateRequirement
appId	AN-01-00-002	Mapper Class	per.per.mint.database.mapper.an.RequirementMapper
Request Object	Requirement		
Response Object	Requirement		
설명	1.요건을 수정 한다.		

3.복합유형 정의서

Requirement(인터페이스요건)

필드ID	필드명	유형	NOT NULL
requirementId	요건ID	String	O
requirementNm	요건명	String	O
status	상태	String	O
statusNm	상태명	String	O
business	업무	Complex<Business>	O
comments	설명	String	
interface	인터페이스	Complex<Interface>	
devExpYmd	개발예상일	String	
devFinYmd	개발완료일	String	
testExpYmd	테스트예정일	String	
testFinYmd	테스트완료일	String	
realExpYmd	운영반영예정일	String	
realFinYmd	운영반영완료일	String	
commentList	인터페이스요건설명 리스트	List<RequirementComment>	
attatchFileList	첨부파일리스트	List<RequirementAttatchFile>	
regUser	등록유저	Complex<User>	
delYn	삭제여부	String	O
regDate	등록일	String	O
regId	등록자	String	O
modDate	수정일	String	
modId	수정자	String	