팔레트 개발 가이드 0.2.0 (교육용)

제개정이력

버전	개정내용	개정일자	작성자
0.1.0	초안작성	2021-07-22	이일용
0.2.0	제개정이력 추가 및 pdf 페이징 처리	2021-07-22	이동욱

용어

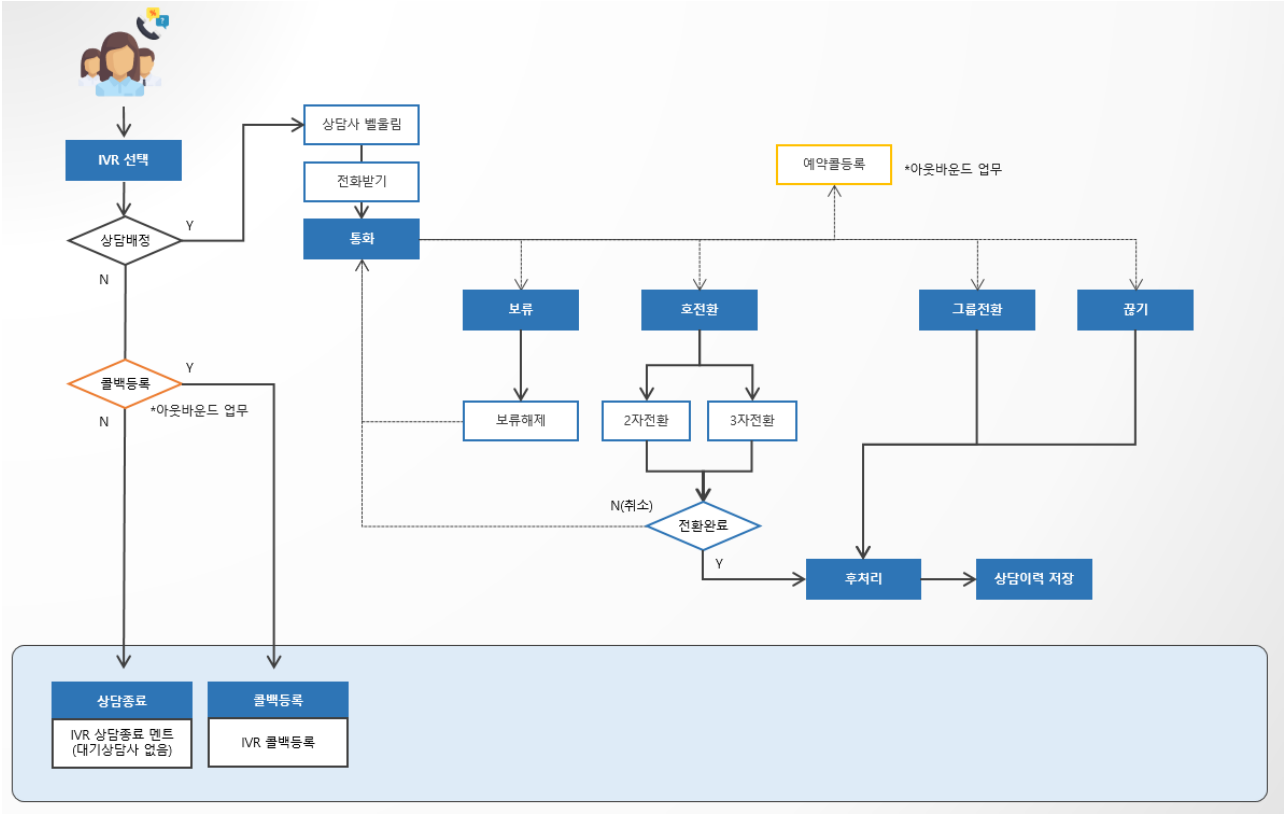
- FE: FrontEnd
- BE: BackEnd

목차

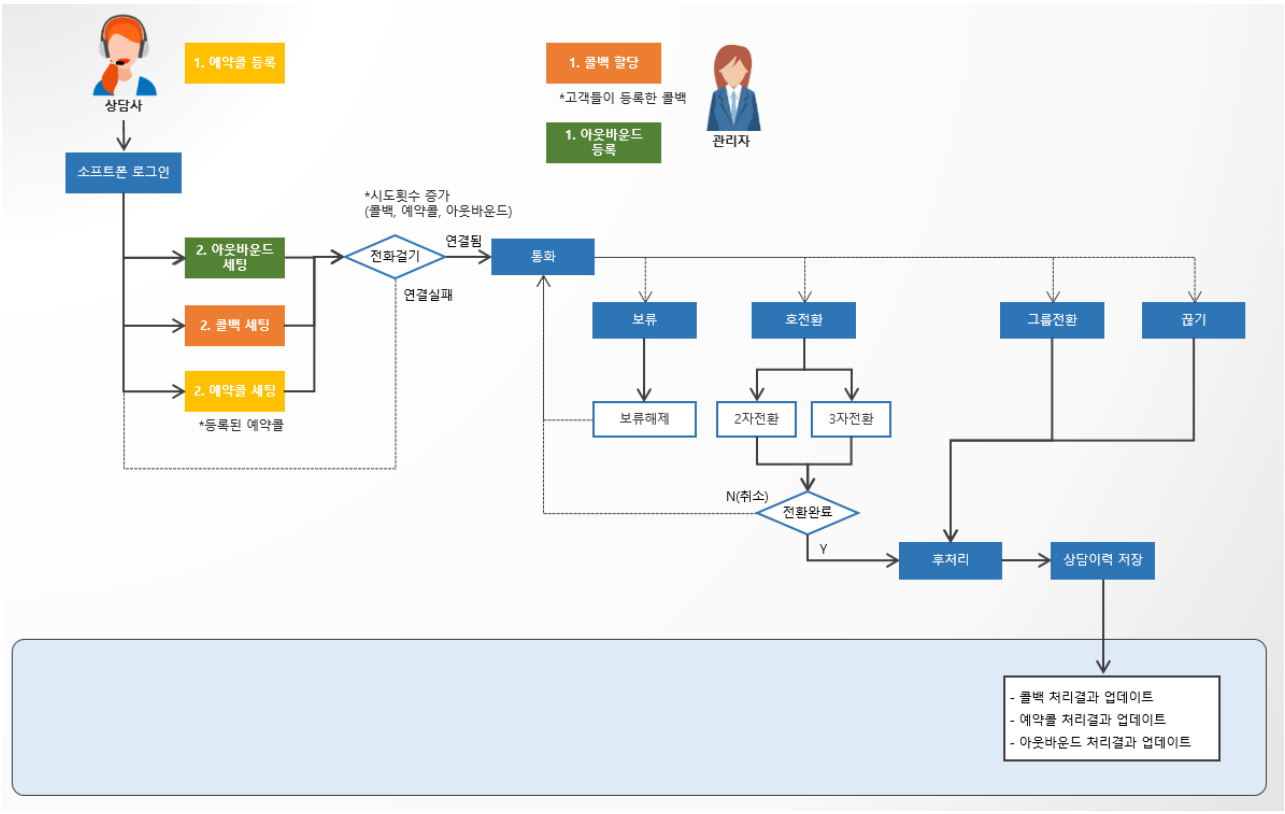
1. 주요 업무 흐름

1.1. 전화 업무 흐름

- 인바운드 흐름

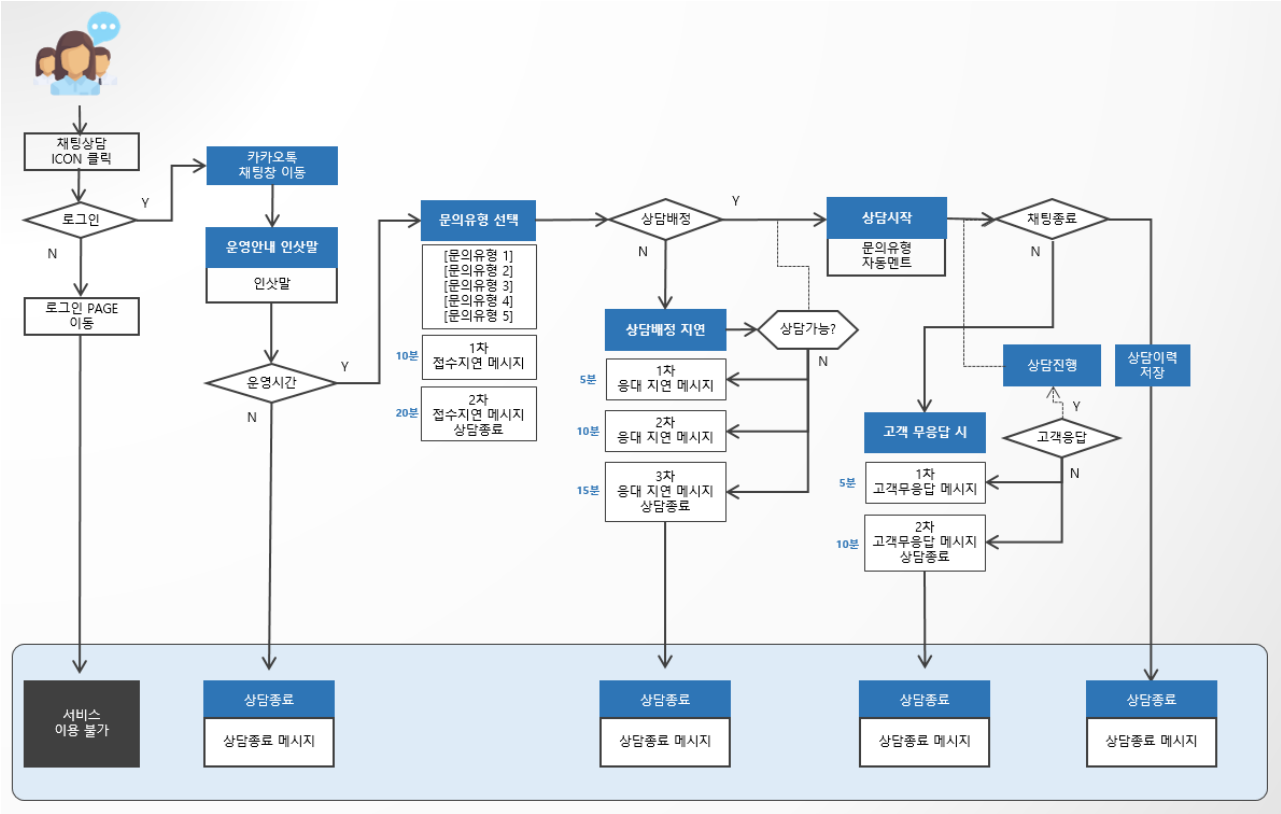


- 콜백/예약콜/캠페인 아웃바운드 흐름

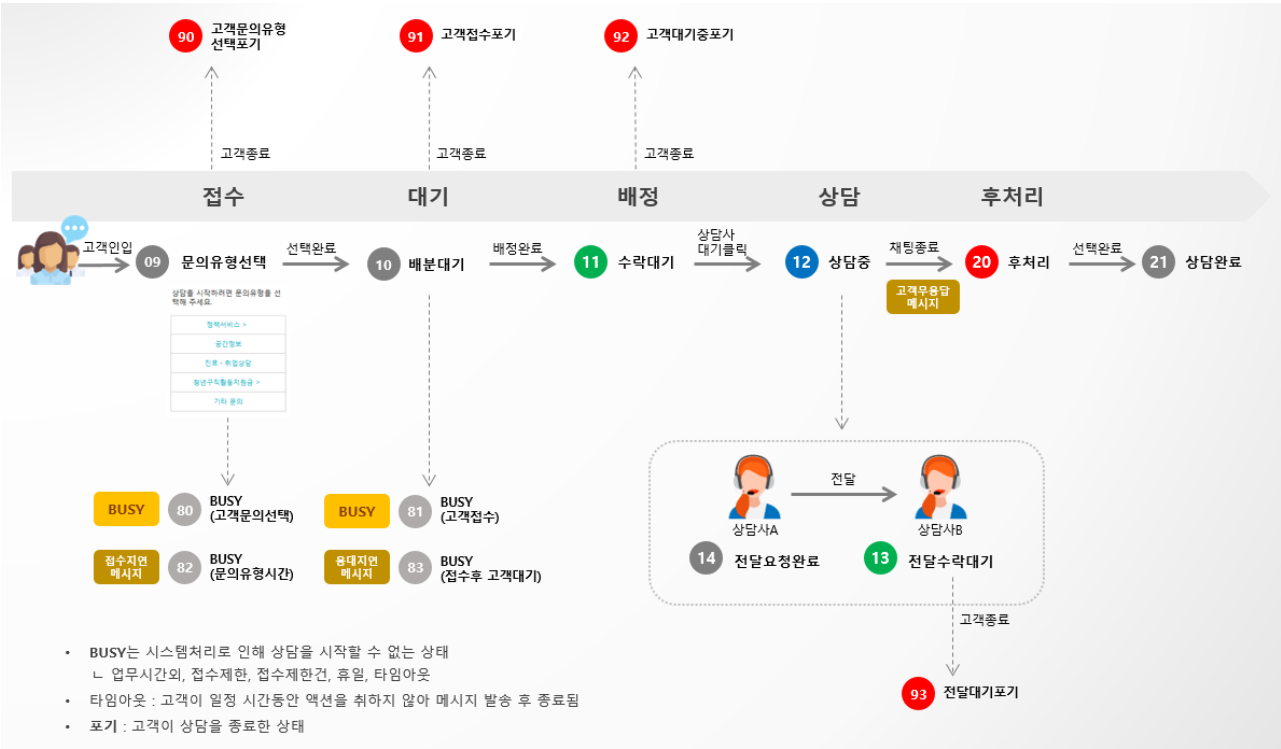


1.2. 채팅 업무 흐름

- 채팅상담 흐름



- 채팅상담 (상태별) 흐름



1.3. 지식 업무 흐름

- 지식 흐름

2. 주요 기술

2.1. 기술 SPEC

- 언어 : 자바
- JVM : openjdk 1.8
- 개발툴 : Spring Tool Suite 4, VS Code
- BE :
    - spring-boot 2.3
        - web, cache, thymleaf, security, websocket, jpa, redis, yaml, undertow
    - mybatis-spring-boot
    - egovframework-idgen
    - jxl, poi
- FE :
    - jquery, vue.js, dhtmlx, ckeditor, json
    - stomp-websocket
    - nodejs
    - router, axios, vue, vuetify
    - html 5, css 3
- DB :
    - redis 5 (linux만 지원됨)
    - oracle 12c, oracle 19c
- 형상관리
    - git
- 빌드 툴 :
    - gradle
- OS :
    - windows 10, linux(Centos)

3. 역할 및 규칙

3.1. 계층별 규칙

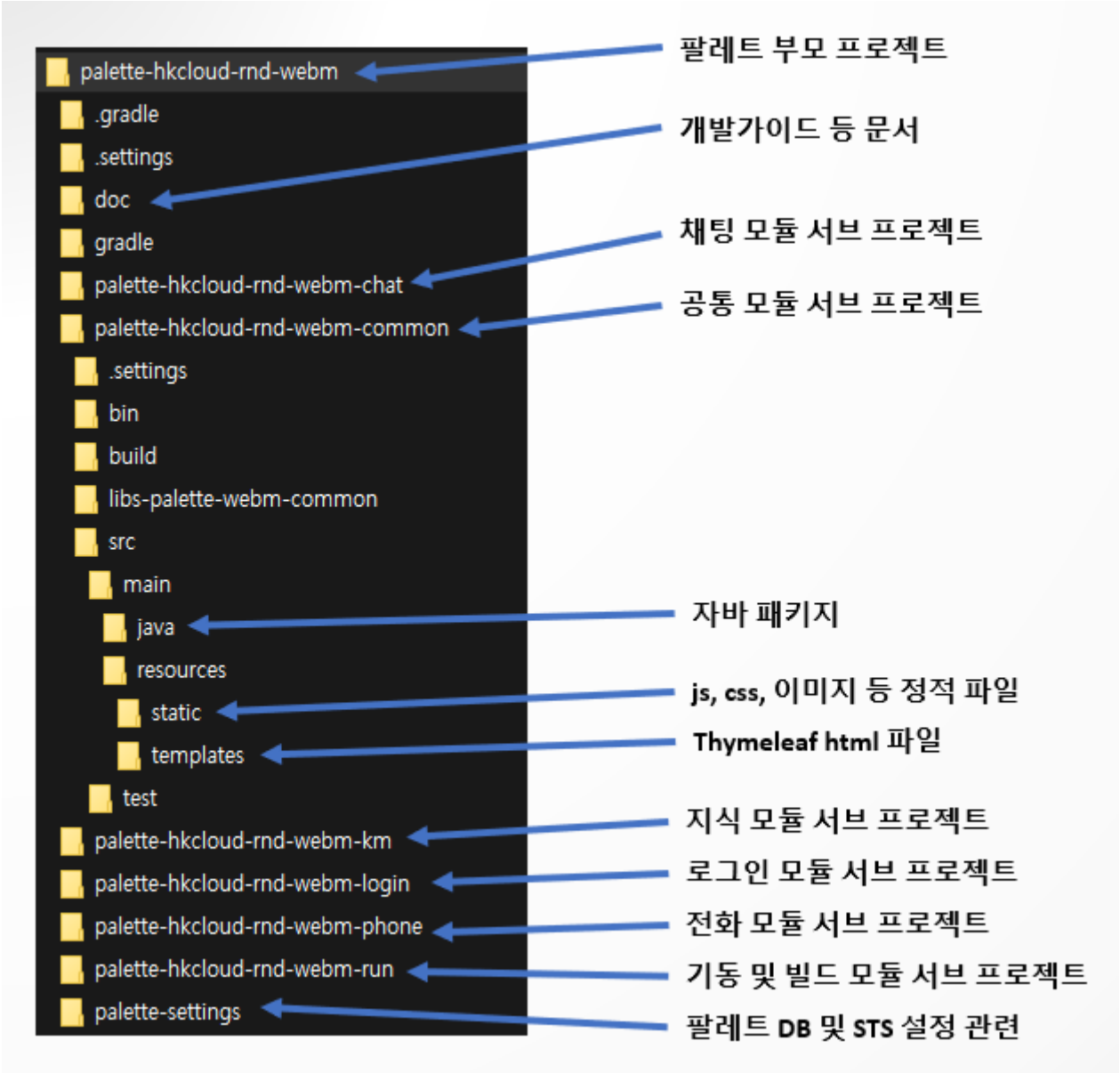
- 프리젠테이션 계층
  - UI (js,html)
    - 클라이언트의 화면과 기능을 책임진다.
  - 컨트롤러 (web, api 패키지)
    - {RULE} web 패키지 클래스명은 *Controller로 끝난다.
    - {RULE} web 패키지는 @Controller 어노테이션을 선언하며 이름은 지정하지 않는다.
    - {RULE} web 패키지는 HTML을 호출한다.
    - {RULE} api 패키지 클래스명은 *RestController로 끝난다.
    - {RULE} api 패키지는 @RestController 어노테이션을 선언하며 이름은 지정하지 않는다.
    - {RULE} api 패키지는 AJAX로 호출하며 JSON을 사용한다.
    - 사용자에게 진입점을 제공하여 BE로의 진입을 책임진다.
    - 사용자에게서 넘어온 데이터를 책임진다.
    - 데이터 유효성 검사를 책임진다.
    - 서비스에 전달할 데이터를 책임진다.
    - 컨트롤러는 서비스와 유틸만 호출한다.
- 비지니스 서비스 계층 (app 패키지)
  - {RULE} app 패키지는 인터페이스를 만들고 인터페이스를 구현한다.
  - {RULE} app 패키지 인터페이스 클래스명은 *Service로 끝난다.
  - {RULE} app 패키지 인터페이스 구현 클래스명은 *ServiceImpl로 끝난다.
  - {RULE} 인터페이스 구현 클래스에 @Sevice({Bean이름}) 어노테이션을 선언한다.
    - {Bean이름}은 인터페이스명이며 첫문자는 소문자를 사용한다.
    - 예) AbcService, AbcServiceImpl
      - @Service("abcService")
  - 서비스는 비지니스 로직 처리에만 충실한다.
  - 비지니스 로직을 책임진다.
  - 트랜젝션을 잭임진다.
  - 서비스는 서비스 또는 DAO/유틸만 호출한다.
- 퍼시스턴트 계층 (dao, repository 패키지)
  - {RULE} dao 패키지는 Mybatice 매퍼 인터페이스를 만든다. (사용하지 않더라도...)
  - {RULE} Mybatice 매퍼 인터페이스 클래스명은 *Mapper로 끝난다.
  - {RULE} 매퍼로 쿼리를 관리하는 경우 dao/provider 패키지 하위에 SQL 쿼리 생성 자바를 둔다.
  - {RULE} SQL 쿼리 생성 자바 클래스명은 *{DB_VENDOR}SqlProvider로 끝난다.
    - {DB_VENDOR}는 Oracle, Mysql 등의 DB 벤더명임.
  - {RULE} 매퍼로 쿼리를 관리하는 경우 dao/enumer 패키지 하위에 디비 테이블 관련 ENUM을 둔다.
  - {RULE} 디비 테이블 관련 ENUM 파일명은 모두 대문자에 _(언더스코어)를 사용한다.
    - 예) FILE_DB_MNG
  - {RULE} 디비 테이블 관련 ENUM 구조는 아래 경로의 ENUM을 참고하여 생성한다.
    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/dao/enumer/FILE_DB_MNG.java
  - {RULE} XML로 쿼리를 관리하는 경우 dao/xml 패키지 하위에 XML 파일을 둔다.
  - {RULE} XML 파일명은 *Mapper_{DB_VENDOR}로 만든다.

        - {DB_VENDOR}는 Oracle, Mysql 등의 DB 벤더명임.
    - 서비스에 필요한 데이터(DB 데이터,XML 등)를 책임진다.
    - 쿼리를 책임진다.
- 파운데이션 계층 (기능 및 유틸, util 패키지)
    - {RULE} util 패키지 클래스명은 *Utils 또는 *Validator로 끝난다.
    - {RULE} 클래스에 @Component 어노테이션을 선언한다.
    - 해당 업무에 유연성을 제공한다.
    - 각 계층에서 공통적으로(또는 예외적으로) 처리가 필요한 경우 유틸을 고려할 수 있다.
        - 예1) 다른 계층에서 공통적으로 사용될 수 있는 경우
            - UUID 생성 로직, 규칙, 유효성 등
        - 예2) 진입점이 다른 경우
            - 파일 처리와 같이 서비스의 역할은 같은데 FE에서 진입하는 경우와 BE에서 진입하는 경우가 있는 경우
                - FE에서의 진입은 컨트롤러가 처리하고, BE에서의 진입은 유틸로 처리할 수 있다.

4. 구조

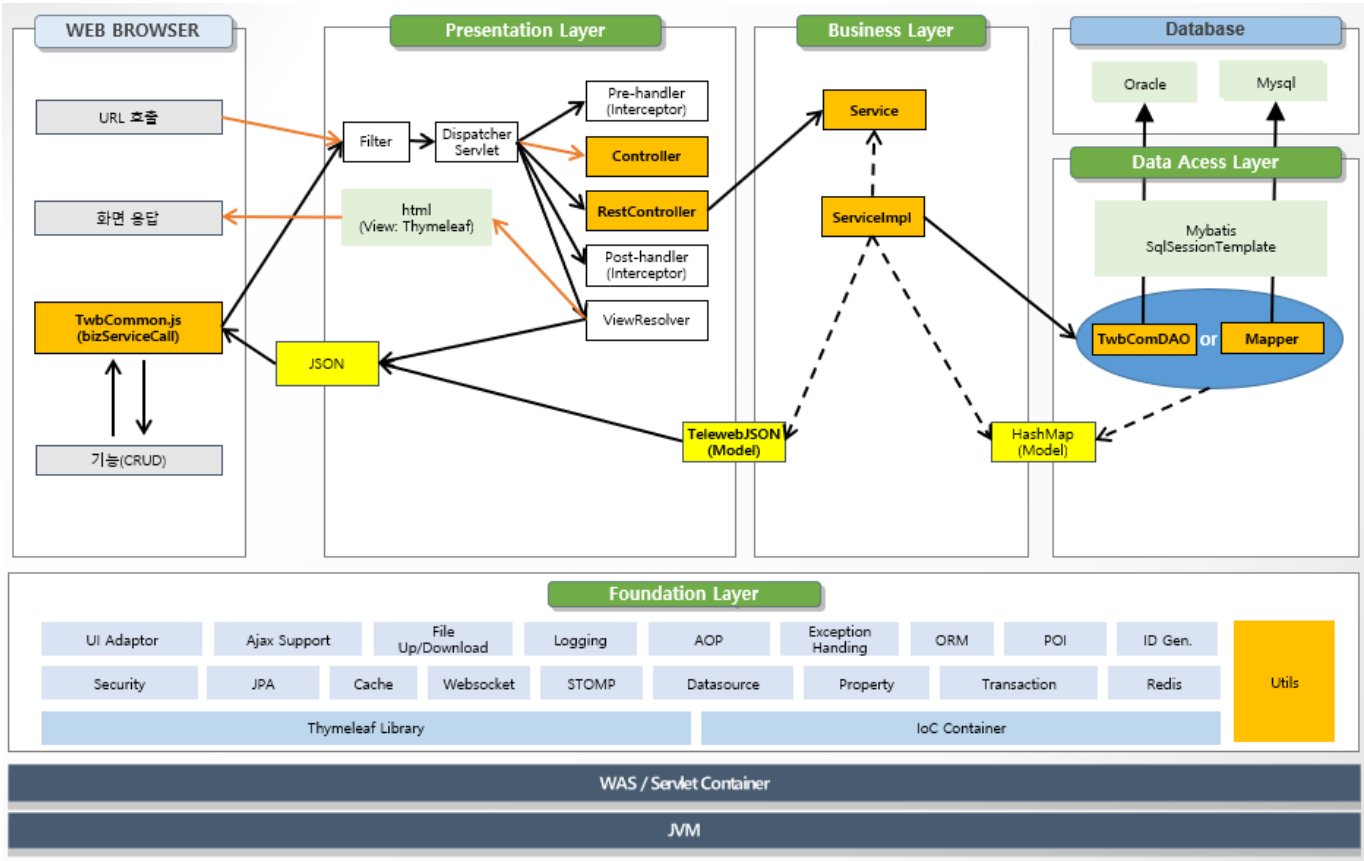
4.1. Project 구조

- gradle 프로젝트 구조 사용



4.2. 자바 패키지 구조

4.3. 개발 프레임워크 아키텍쳐 뷰



5. 주요 기능별

5.1. Spring-boot 속성 및 profile

- springframework는 profile 설정을 통해 환경마다 다른 설정을 호출할 수 있는 기능을 제공한다.
- jvm argument로 spring.profiles.active를 선언하는 방식임

```
-Dspring.profiles.active=local,local-phone,local-chat,local-km
```

- 개발 단계에 따른 모듈별 프로필
  - 로컬 (local)
    - local, local-phone, local-chat, local-km
  - 개발 (dev)
    - dev, dev-phone, dev-chat, dev-km
  - 운영 (production)
    - production, production-phone, production-chat, production-km
- 구조
  - 프로필은 spring.profiles 와 spring.profiles.include 가 사용됨
  - 속성은 YAML 사용됨
    - YAML 문법 참고 - YAML Syntax
  - spring.profiles.active 선언된 프로필에 따라 yml을 찾는다.
    - local (로컬공통프로필)
      - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml

```
spring.profiles:
    -local
spring.profiles.include:
    -common
    -logging
    -server
    -redis
    -datasources
    -jpa
    -bbs
    -editor
    -excel
    -file
    -alimtalk
    -proxy
    -palette
---
```

```
  spring.config.activate:
      on-profile: "common"
  ...
  ---
  spring.config.activate:
      on-profile: "logging"
  ...
  ########################################################
  #
  # 팔레트(palette) 속성
  # --------------------------------------------------
  # include 'palette' profile
  ########################################################
  ---
  spring.config.activate:
      on-profile: "palette"
  # ---------------------------------------
  # PaletteProperties
  # ---------------------------------------
  palette:
      enabled: true
      ...
```

- local-phone 로컬전화프로필
  - ~/palette-hkcloud-rnd-webm-phone/src/main/resources/application-local-phone.yml

```
spring.profiles:
  -local-phone

spring.profiles.include:
  -phone

########################################################
#
# 전화(phone) 속성
# --------------------------------------------------
# include 'phone' profile
########################################################
---
spring.config.activate:
  on-profile: "phone"
  # ---------------------------------------
  # PhoneProperties
  # ---------------------------------------
  phone:
    enabled: true
    ...
```

- local-chat 로컬채팅프로필
  - ~/palette-hkcloud-rnd-webm-chat/src/main/resources/application-local-chat.yml

```
spring.profiles:
  -local-chat

spring.profiles.include:
  -chat

##################################################
#
# 채팅(chat) 속성
# ------------------------------------------------
# include 'chat' profile
##################################################
---
spring.config.activate:
  on-profile: "chat"
  # ---------------------------------------
  # ChatProperties
  # ---------------------------------------
  chat:
    enabled: true
    ...
```

- local-km 로컬지식프로필
  - ~/palette-hkcloud-rnd-webm-km/src/main/resources/application-local-km.yml

```
spring.profiles:
  -local-km

spring.profiles.include:
  -km

##################################################
#
# 지식(km) 속성
# ------------------------------------------------
# include 'km' profile
##################################################
---
spring.config.activate:
  on-profile: "km"
  # ---------------------------------------
  # KmProperties
  # ---------------------------------------
  km:
    enabled: true
    ...
```

- 자바 CONFIG
  - 각 profile include에 상응하는 Properties java가 존재함
  - Spring boot 기동 시 유효성을 체크함
  - 공통
    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/properties 각 하위 패키지
  - 전화
    - ~/palette-hkcloud-rnd-webm-phone/src/main/java/kr/co/hkcloud/palette/config/properties/phone/PhoneProperties.java
  - 채팅
    - /palette-hkcloud-rnd-webm-chat/src/main/java/kr/co/hkcloud/palette/config/properties/chat/ChatProperties.java
  - 지식
    - /palette-hkcloud-rnd-webm-km/src/main/java/kr/co/hkcloud/palette/config/properties/km/KmProperties.java
  - 해당 profile include에 상응하는 java는 yml 속성과 1:1 매핑되어 있으며, 해당 자바 데이터 형을 사용함(권장)
    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml 중 profiles include: **editor**

```yaml
    ######################################################
    #
    # 에디터(editor) 속성
    # ----------------------------------------------------
    # include 'editor' profile
    ######################################################
    ---
    spring.config.activate:
        on-profile: "editor"
    # ----------------------------------------
    # EditorProperties
    # ----------------------------------------
    editor:
        enabled: true
        name: "ckeditor"
        version: "4.7.2"
        repository:
        # ----------------------------------------
        # 루트 저장소
        # ----------------------------------------
        root-dir:
file:///d:/PALETTE/hkcloud/rnd/repository/web/editor

        # ----------------------------------------
        # 에디터 이미지 저장소
        # ----------------------------------------
        km:
```

```
            images:
                target-type: FILE
                enabled: true
                max-files: 1
                max-filesize: 2MB
                accepted-files:
                - "jpg"
                - "png"
                - "gif"
                dir: ${editor.repository.root-dir}/km/images
                temp-dir: ${editor.repository.root-
dir}/km/images/temp
        bbs:
            images:
                target-type: FILE
                enabled: true
                max-files: 1
                max-filesize: 2MB
                accepted-files:
                - "jpg"
                - "png"
                - "gif"
                dir: ${editor.repository.root-dir}/bbs/images
                temp-dir: ${editor.repository.root-
dir}/bbs/images/temp
```

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/properties/editor/EditorProperties.java

```
/**
* 에디터(editor) 속성
*
* @author 이일용
*
*/
@Getter
@RequiredArgsConstructor
@Validated
@ConstructorBinding
@ConfigurationProperties(prefix = "editor")
public class EditorProperties
{
    @NotNull
    private final boolean enabled;

    @NotNull
    private final PaletteEditor name;  //이름

    @NotBlank
    private final String version; //버전
```

```
    private final Repository repository;

    @Getter
    @RequiredArgsConstructor
    public static final class Repository
    {
        @NotNull
        private final File rootDir;

        private final Km  km;
        private final Bbs bbs;

        @Getter
        @RequiredArgsConstructor
        public static final class Km
        {
            private final Images images;

            @Getter
            @RequiredArgsConstructor
            public static final class Images
            {
                @NotNull
                private final RepositoryTargetType
targetType;
                @NotNull
                private final boolean              enabled;
                @NotNull
                private final int                  maxFiles;
                @DataSizeUnit(DataUnit.MEGABYTES)
                @NotNull
                private final DataSize
maxFilesize;
                @NotNull
                private final List<String>
acceptedFiles;
                @NotNull
                private final Path                 dir;
                @NotNull
                private final Path                 tempDir;
            }
        }

        @Getter
        @RequiredArgsConstructor
        public static final class Bbs
        {
            private final Images images;

            @Getter
            @RequiredArgsConstructor
            public static final class Images
            {
```

```
                @NotNull
                private final RepositoryTargetType
targetType;
                @NotNull
                private final boolean              enabled;
                @NotNull
                private final int                  maxFiles;
                @DataSizeUnit(DataUnit.MEGABYTES)
                @NotNull
                private final DataSize
maxFilesize;
                @NotNull
                private final List<String>
acceptedFiles;
                @NotNull
                private final Path                 dir;
                @NotNull
                private final Path                 tempDir;
            }
        }
    }
}
```

- 프로필 사용법
  - 현재 활성중인 프로필 확인 (local, dev, production)
    - 컴포넌트에 주입

```
private final PaletteProfileUtils paletteProfileUtils;
```

    - 함수에서 사용

```
PaletteProfiles profile = paletteProfileUtils.getActiveProfile();
```

  - 현재 활성중인 모듈 확인 (phone, chat, km)
    - 컴포넌트에 주입

```
private final PaletteProfileUtils paletteProfileUtils;
```

    - 함수에서 사용

```
List<PaletteModules> modules =
paletteProfileUtils.getActiveModule();
```

- 속성 사용법
    - 에디터 속성 중 게시판 이미지의 허용된 확장자는?
        - 컴포넌트에 주입

```
private final EditorProperties editorProperties;
```

        - 함수에서 사용

```
final List<String> acceptedFiles =
editorProperties.getBbs().getImages().getAcceptedFiles();
```

- polling 관련 기능 활성화 여부
    - 개발 편의성을 위해 폴링 기능은 활성 여부에 따라 on/off 되도록 함
    - 개발/운영(dev/production) 프로필에서는 활성화되어 있어야 정상 작동하므로, local에서만 사용할 것을 권장함.
    - 관련 설정
        - 각 모듈의 application*.yml
        - 실시간 공지사항 활성화 여부 추가됨

```
main-notice-enabled: false
```

        - 전화메인 예약콜/전광판 활성화 여부

```
main-resve-call-enabled: false
main-check-board-enabled: false
```

        - 채팅 라우터 활성 여부

```
router-enabled: false
```

- 커스텀 수정 시 주의 사항
    - module마다 동일한 이름의 application-{profile}.properties를 선언하여 중복 호출을 할 수 없다.
    - 출처 : 파란하늘의 지식창고

5.2. Datasource (Mybatis)

- 구조
  - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml

```
#######################################################
#
# 데이터소스(datasources) 속성
# ㄴ명명 규칙: datasources.[시스템명].[master or slave].[각 속성]
# -----------------------------------------------
# include 'datasources' profile
#######################################################
---
spring.config.activate:
  on-profile: "datasources"
# ---------------------------------------
# DatasourcesProperties
# ---------------------------------------
datasources:
  datasource:
    default-pool-name: "palette"
    palette:
      db-vendor: "Oracle"
      master:
        jndi-name: "PALETTE_MASTER_DS"
        expected-type: javax.sql.Datasource
        jdbc-url:
ENC(dq2iz3tcpVULlmLIZHGENQ0hFFj0a7WV2SLqeqVgNrUrXX3GbOrsE3fXGYzPpYhGjfj
SB4HQmsy7zapyHI8d4C8gKSedJgUls5m5wEWhwRY=)
        username:
ENC(5inANxK1AMUl4G8JU/1tM+zf8yDlprK90+CI+ZXAvDf89EKZZniYoo1nceWwtnc0)
        password:
ENC(vgus9OimxxxBJW+YU7e69Y1LrufVRvEJNzuyA5hGKU1wTC5cTMiTKMZZ1c33dXdE)
        #driver-class-name: "oracle.jdbc.driver.OracleDriver"
        driver-class-name: "net.sf.log4jdbc.sql.jdbcapi.DriverSpy"
        type: com.zaxxer.hikari.HikariDataSource
        hikari:
          maximum-pool-size: 20
          minimum-idle: 10
          connection-test-query: SELECT 1 FROM DUAL
          connection-timeout: 30000
      slave:
        jndi-name: "PALETTE_SLAVE_DS"
        expected-type: javax.sql.Datasource
        jdbc-url:
ENC(dq2iz3tcpVULlmLIZHGENQ0hFFj0a7WV2SLqeqVgNrUrXX3GbOrsE3fXGYzPpYhGjfj
SB4HQmsy7zapyHI8d4C8gKSedJgUls5m5wEWhwRY=)
        username:
ENC(5inANxK1AMUl4G8JU/1tM+zf8yDlprK90+CI+ZXAvDf89EKZZniYoo1nceWwtnc0)
        password:
ENC(vgus9OimxxxBJW+YU7e69Y1LrufVRvEJNzuyA5hGKU1wTC5cTMiTKMZZ1c33dXdE)
```

```
        #driver-class-name: "oracle.jdbc.driver.OracleDriver"
        driver-class-name: "net.sf.log4jdbc.sql.jdbcapi.DriverSpy"
        type: com.zaxxer.hikari.HikariDataSource
        hikari:
          maximum-pool-size: 20
          minimum-idle: 10
          connection-test-query: SELECT 1 FROM DUAL
          connection-timeout: 30000
# ----------------------------------------
# TransactionProperties
# ----------------------------------------
spring.transaction:
  # Default transaction timeout. If a duration suffix is not specified,
seconds will be used.
  default-timeout: 60s
  # Whether to roll back on commit failures.
  #rollback-on-commit-failure:
```

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/properties/datasources/DatasourcesProperties.java

```
/**
 * 데이터소스 팔레트 속성
 * @author RND
 */
@Getter
@RequiredArgsConstructor
@Validated
@ConstructorBinding
@ConfigurationProperties(prefix = "datasources")
public class DatasourcesProperties
{
    private final Datasource datasource;

    //데이터소스 속성
    @Getter
    @RequiredArgsConstructor
    public static final class Datasource
    {
        @NotNull
        private final DatasourcePoolName defaultPoolName;

        //팔레트 데이터소스
        private final Palette palette;

        @Getter
        @RequiredArgsConstructor
        public static final class Palette
        {
            @NotNull
```

```
            private final DatasourceDbVendor dbVendor;

            private final Master master;
            private final Slave slave;

            //Master 속성
            @Getter
            @RequiredArgsConstructor
            public static final class Master
            {
                @NotBlank
                private final String jndiName;
            }

            //Slave 속성
            @Getter
            @RequiredArgsConstructor
            public static final class Slave
            {
                @NotBlank
                private final String jndiName;
            }
        }
    }
}
```

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/datasources/datasource/palette/PaletteRoutingDataSourceConfig.java

```
/**
 * 팔레트 라우팅 데이터소스 설정 - JNDI 지원
 * @author 이일용
 */
@Slf4j
@Configuration
@RequiredArgsConstructor
@MapperScan(value = "kr.co.hkcloud.palette",
            annotationClass = PaletteConnMapper.class,
            sqlSessionFactoryRef = "paletteRoutingSqlSessionFactory")
@EnableTransactionManagement
public class PaletteRoutingDataSourceConfig
{
    private final DatasourcesProperties     datasourceProperties;
    private final VendorDatabaseIdProperies vendorDatabaseIdProperies;
    private final PaletteProfileUtils       paletteProfileUtils;
    private JndiDataSourceLookup            lookup = new
JndiDataSourceLookup();

    /**
    * Master 데이터소스 -JNDI를 사용하고자 하면 해당 개발단계 profile을 수
```

```
정하여 적용할 것
    * @return DataSource
    */
    @Primary
    @Bean(destroyMethod = "close")
    @ConfigurationProperties(prefix =
"datasources.datasource.palette.master")
    public DataSource paletteRoutingMasterDataSource()
    {
        final PaletteProfiles profile =
paletteProfileUtils.getActiveProfile();
        log.trace("*palette master datasource profile={}", profile);
        switch(profile)
        {
            case local:
            case dev:
            case production:
            {
                //HikariDataSource
                return
DataSourceBuilder.create().type(HikariDataSource.class).build();
            }
            case uat:
            {
                //JNDI를 사용하는 경우 사용
                return
lookup.getDataSource(datasourceProperties.getDatasource().getPalette().
getMaster().getJndiName());
            }
            default:
            {
                return null;
            }
        }
    }

    /**
    * Slave 데이터소스 -JNDI를 사용하고자 하면 해당 개발단계 profile을 수정
하여 적용할 것
    * @return DataSource
    */
    @Bean(destroyMethod = "close")
    @ConfigurationProperties(prefix =
"datasources.datasource.palette.slave")
    public DataSource paletteRoutingSlaveDataSource()
    {
        PaletteProfiles profile =
paletteProfileUtils.getActiveProfile();
        log.trace("*palette slave datasource profile={}", profile);
        switch(profile)
        {
            case local:
            case dev:
            case production:
```

```
            {
                //HikariDataSource
                return
DataSourceBuilder.create().type(HikariDataSource.class).build();
            }
            case uat:
            {
                //JNDI를 사용하는 경우 사용
                return
lookup.getDataSource(datasourceProperties.getDatasource().getPalette().
getSlave().getJndiName());
            }
            default:
            {
                return null;
            }
        }
    }

    /**
    * @param  routingMasterDataSource
    * @param  routingSlaveDataSource
    * @return
    */
    @Bean
    public DataSource paletteRoutingDataSource()
    {
        Map<Object, Object> targetDataSourceMap = new
LinkedHashMap<Object, Object>();
        targetDataSourceMap.put("master",
paletteRoutingMasterDataSource());
        targetDataSourceMap.put("slave_1",
paletteRoutingSlaveDataSource());
        PaletteReplicationRoutingDataSource routingDataSource = new
PaletteReplicationRoutingDataSource();
        routingDataSource.setTargetDataSources(targetDataSourceMap);

routingDataSource.setDefaultTargetDataSource(paletteRoutingMasterDataSo
urce());
        return routingDataSource;
    }
    ...

    /**
    * @param  routingLazyDataSource
    * @param  applicationContext
    * @return
    * @throws Throwable
    */
    @Bean
    public SqlSessionFactory
paletteRoutingSqlSessionFactory(ApplicationContext applicationContext)
throws Throwable
    {
```

```
        String locationPattern = "";

        DatabaseIdProvider vendorDatabaseIdProvider = new
VendorDatabaseIdProvider();

vendorDatabaseIdProvider.setProperties(vendorDatabaseIdProperies.getDbV
endorProperties());

        DatasourceDbVendor dbVendor =
DatasourceDbVendor.valueOf(vendorDatabaseIdProvider.getDatabaseId(palet
teRoutingLazyDataSource()));
        log.debug("dbVendor=============> {}", dbVendor);
        switch(dbVendor)
        {
            case Oracle:
            {
                locationPattern =
"classpath*:kr/co/hkcloud/palette/**/dao/xml/*Mapper_Oracle.xml";
                break;
            }
            case Mysql:
            {
                locationPattern =
"classpath*:kr/co/hkcloud/palette/**/dao/xml/*Mapper_Mysql.xml";
                break;
            }
            default:
            {
                locationPattern =
"classpath*:kr/co/hkcloud/palette/**/dao/xml/*Mapper_Mysql.xml";
                break;
            }
        }

        final SqlSessionFactoryBean sessionFactory = new
SqlSessionFactoryBean();
        sessionFactory.setDataSource(paletteRoutingLazyDataSource());
        sessionFactory.setTypeHandlersPackage("kr.co.hkcloud.palette");
        sessionFactory.setTypeAliasesPackage("kr.co.hkcloud.palette");
        sessionFactory.setDatabaseIdProvider(vendorDatabaseIdProvider);

sessionFactory.setMapperLocations(applicationContext.getResources(locat
ionPattern));
        return sessionFactory.getObject();
    }


    /**
    * @param  paletteRoutingSqlSessionFactory
    * @return
    */
    @Primary
    @Bean
    public SqlSessionTemplate
```

```
paletteRoutingSqlSessionTemplate(@Qualifier("paletteRoutingSqlSessionFa
ctory") SqlSessionFactory paletteRoutingSqlSessionFactory)
    {
        org.apache.ibatis.session.Configuration mybatisConfig =
paletteRoutingSqlSessionFactory.getConfiguration();

        ...

        return new SqlSessionTemplate(paletteRoutingSqlSessionFactory);
    }


    @Bean
    public PlatformTransactionManager transactionManager()
    {
        return new
DataSourceTransactionManager(paletteRoutingLazyDataSource());
    }
}
```

- TwbComDAO 사용하는 방식
  - 흐름
    - 컨트롤러 -> 서비스 -> TwbComDAO (Mybatis SqlSession) -> XML
  - 서비스단에서 아래와 같이 호출

```
@Service("boardService")
public class BoardServiceImpl implements BoardService
{
    private final TwbComDAO twbComDAO;

    @Override
    public TelewebJSON selectRtnSystemNotice(TelewebJSON
jsonParams) throws TelewebAppException
    {
        return
twbComDAO.select("kr.co.hkcloud.palette.board.dao.BoardMapper",
"selectRtnSystemNotice", jsonParams);
    }
    ...
}
```

  - TwbComDAO에서 아래와 같이 처리

```
@Repository
@Transactional
public class TwbComDAO
{
    private SqlSession paletteRoutingSqlSessionTemplate;
```

```
    @Autowired
    @Qualifier("paletteRoutingSqlSessionTemplate")
    public void setPaletteRoutingSqlSessionTemplate(SqlSession
paletteRoutingSqlSessionTemplate) {
        this.paletteRoutingSqlSessionTemplate =
paletteRoutingSqlSessionTemplate;
    }
    ...

    public TelewebJSON select(String sqlNameSpace, String sqlNm,
TelewebJSON acJson) throws TelewebDaoException {
        ...
        //전달 받은 POOL이 있는지 체크
        if(obj.containsKey("POOL_NM")) {
            //있으면 해당 POOL을 사용
            DatasourcePoolName datasourcePoolName =
DatasourcePoolName.valueOf(obj.getString("POOL_NM"));
            return this.select(sqlNameSpace, sqlNm, acJson, null,
datasourcePoolName);
        }
        else {
            //없으면 기본 데이터소스를 사용함 (default-pool-name)
            return this.select(sqlNameSpace, sqlNm, acJson, null,
DatasourcePoolName.palette);
        }
    }
    ...

    public TelewebJSON select(String sqlNameSpace, String sqlNm,
TelewebJSON acJson, HttpServletRequest objRequest,
DatasourcePoolName datasourcePoolName) throws TelewebDaoException
    {
        ...
        String strQry = sqlNameSpace + sqlNm;

        retMap = this.select(strQry, map, objRequest,
datasourcePoolName);

        objRetJson.setDataObject(getJsonArrayFromList(retMap));
        ...
        return objRetJson;
    }
    ...

    /**
     * 리스트 조회
     */
    private List<HashMap<String, Object>> select(String id,
HashMap<String, Object> map, HttpServletRequest objRequest,
DatasourcePoolName datasourcePoolName) throws TelewebDaoException
    {
        return getSqlSession(objRequest,
datasourcePoolName).selectList(id, map);
    }
```

```
    ...

    /**
    * sqlSession을 변환한다.
    */
    private SqlSession getSqlSession(HttpServletRequest
objRequest, DatasourcePoolName datasourcePoolName) throws
TelewebDaoException
    {
        SqlSession sqlSession = null;
        switch(datasourcePoolName)
        {
            case palette:
                sqlSession =
this.paletteRoutingSqlSessionTemplate;
                break;
        }
        return sqlSession;
    }
    ...
}
```

- 해당 XML 호출

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="kr.co.hkcloud.palette.board.dao.BoardMapper">
...

<!-- 시스템공지사항 데이터 조회 -->
<select id="selectRtnSystemNotice"  parameterType=
"java.util.HashMap" resultType="java.util.HashMap">
<![CDATA[
SELECT  P1.*
FROM (
      SELECT
          ...
    ) P1
]]>
</select>
...

</mapper>
```

- 어노테이션을 사용하는 방식
  - 흐름
    - 컨트롤러 -> 서비스 -> DAO Mapper 인터페이스 -> SQLProvider (Non XML)
  - 서비스단에서 아래와 같이 호출

```
@Service("fileDbMngService")
public class FileDbMngServiceImpl implements FileDbMngService
{
    ...
    private final FileDbMngMapper fileDbMngMapper;
    ...

    @Override
    public FileDbMngSelectTargetTypeResponse
selectRepositoryTargetType(final FileDbMngSelectRequest
fileDbMngSelectRequest) throws PaletteAppException
    {
        return
fileDbMngMapper.selectRepositoryTargetType(fileDbMngSelectRequest)
;
    }
}
```

  - DAO Mapper에서 아래와 같이 처리

```
@PaletteConnMapper
public interface FileDbMngMapper
{
    ...

    /**
      * 저장소대상유형 조회
      *
      * @param  fileDbMngSelectRequest
      * @return
      * @throws PaletteDaoException
      */
      @SelectProvider(value =
FileDbMngSelectOracleSqlProvider.class,
                      method = "selectRepositoryTargetType",
                      databaseId = "Oracle")
      @SelectProvider(value =
FileDbMngSelectMysqlSqlProvider.class,
                      method = "selectRepositoryTargetType",
                      databaseId = "Mysql")
      @Valid
      FileDbMngSelectTargetTypeResponse
selectRepositoryTargetType(@Validated(FileDbMngRequest.GroupServic
eSelectTargetType.class) final FileDbMngSelectRequest
fileDbMngSelectRequest) throws PaletteDaoException;
```

```
        ...
}
```

- DAO SQL Provider 로 쿼리문을 생성

```
  public class FileDbMngSelectOracleSqlProvider implements
ProviderMethodResolver
  {
      /**
      * 저장소대상유형 조회
      * @param fileDbMngSelectRequest
      * @return
      */
      public static String selectRepositoryTargetType(final
FileDbMngSelectRequest fileDbMngSelectRequest)
      {
          return new SQL() {{
              SELECT(

FILE_DB_MNG.COLUMN.PLT_FILE.ASP_CUST_KEY.name()
//-- ASP고객키
                  ,
FILE_DB_MNG.COLUMN.PLT_FILE.FILE_GROUP_KEY.name()    //AS
fileGroupKey      //-- 파일그룹키
                  , FILE_DB_MNG.COLUMN.PLT_FILE.FILE_KEY.name()
//AS   fileKey           //-- 파일키
                  ,
FILE_DB_MNG.COLUMN.PLT_FILE.FILE_ACCESS_TYPE.name()
//-- 파일엑세스유형(public:공개,private:비공개)
                  , FILE_DB_MNG.COLUMN.PLT_FILE.TARGET_TYPE.name()
//AS   targetType        //-- 저장소대상유형
                  ,
FILE_DB_MNG.COLUMN.PLT_FILE.ORIGINAL_FILENAME.name() //AS
originalFilename  //-- 진짜 파일명
              );
              FROM(
                      FILE_DB_MNG.TABLE.PLT_FILE.name()
              );
              WHERE(
                      String.format("%s = #{aspCustKey}"     ,
FILE_DB_MNG.COLUMN.PLT_FILE.ASP_CUST_KEY.name())
                  , String.format("%s = #{fileGroupKey}"   ,
FILE_DB_MNG.COLUMN.PLT_FILE.FILE_GROUP_KEY.name())
                  , String.format("%s = #{fileKey}"        ,
FILE_DB_MNG.COLUMN.PLT_FILE.FILE_KEY.name())
                  , String.format("%s = #{fileAccessType}" ,
FILE_DB_MNG.COLUMN.PLT_FILE.FILE_ACCESS_TYPE.name())
              );

              //업무유형
```

```
    if(!StringUtils.isEmpty(fileDbMngSelectRequest.getBusiType()))
                {
                    WHERE(String.format("%s = #{busiType}",
    FILE_DB_MNG.COLUMN.PLT_FILE.BUSI_TYPE.name()));
                }
                //저장소 대상 유형

    if(!StringUtils.isEmpty(fileDbMngSelectRequest.getTargetType()))
                {
                    WHERE(String.format("%s = #{targetType}",
    FILE_DB_MNG.COLUMN.PLT_FILE.TARGET_TYPE.name()));
                }
                //경로유형

    if(!StringUtils.isEmpty(fileDbMngSelectRequest.getPathType()))
                {
                    WHERE(String.format("%s = #{pathType}",
    FILE_DB_MNG.COLUMN.PLT_FILE.PATH_TYPE.name()));
                }
                //MIME 유형

    if(!StringUtils.isEmpty(fileDbMngSelectRequest.getMimeType()))
                {
                    WHERE(String.format("%s = #{mimeType}",
    FILE_DB_MNG.COLUMN.PLT_FILE.MIME_TYPE.name()));
                }
            }}.toString();
        }
        ...
    }
```

- 제우스 등의 상용 WAS JNDI 사용하는 경우
  - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-production.yml

```
##########################################################
#
# 데이터소스(datasources) 속성
# ㄴ명명 규칙: datasources.[시스템명].[master or slave].[각 속성]
# --------------------------------------------------
# include 'datasources' profile
##########################################################
---
spring.config.activate:
  on-profile: "datasources"
# ---------------------------------------
# DatasourcesPaletteProperties
# ---------------------------------------
datasources:
  datasource:
    default-pool-name: "palette"
    palette:
```

```
      db-vendor: "Oracle"
      master:
        jndi-name: "PALETTE_MASTER_DS"  #WAS에 설정된 데이터소스 이름
        expected-type: javax.sql.Datasource
      slave:
        jndi-name: "PALETTE_SLAVE_DS"  #WAS에 설정된 데이터소스 이름
        expected-type: javax.sql.Datasource
```

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/datasources/datasource/palette/PaletteRoutingDataSourceConfig.java

```
    /**
    * Master 데이터소스 -JNDI를 사용하고자 하면 해당 개발단계 profile을 수정하여 적용할 것
    *
    * @return DataSource
    */
    @Primary
    @Bean(destroyMethod = "close")
    @ConfigurationProperties(prefix = "datasources.datasource.palette.master")
    public DataSource paletteRoutingMasterDataSource()
    {
        PaletteProfiles profile = paletteProfileUtils.getActiveProfile();
        log.trace("*palette master datasource profile={}", profile);
        switch(profile)
        {
            //로컬은 HikariDataSource
            case local:
            {
                return DataSourceBuilder.create().type(HikariDataSource.class).build();
            }
            //개발 및 운영은 JNDI
            case dev:
            case production:
            {
                return lookup.getDataSource(datasourceProperties.getDatasource().getPalette().getMaster().getJndiName());
            }
            default:
            {
                return null;
            }
        }
    }
```

```
    /**
    * Slave 데이터소스 -JNDI를 사용하고자 하면 해당 개발단계 profile을 수정
하여 적용할 것
    *
    * @return DataSource
    */
    @Bean(destroyMethod = "close")
    @ConfigurationProperties(prefix =
"datasources.datasource.palette.slave")
    public DataSource paletteRoutingSlaveDataSource()
    {
        PaletteProfiles profile =
paletteProfileUtils.getActiveProfile();
        log.trace("*palette slave datasource profile={}", profile);
        switch(profile)
        {
            //로컬은 HikariDataSource
            case local:
            {
                return
DataSourceBuilder.create().type(HikariDataSource.class).build();
            }
            //개발 및 운영은 JNDI
            case dev:
            case production:
            {
                return
lookup.getDataSource(datasourceProperties.getDatasource().getPalette().
getMaster().getJndiName());
            }
            default:
            {
                return null;
            }
        }
    }
```

5.3. Transaction

- {RULE] 서비스단(app 패키지)에서 관리한다.
    - 특히 web/api 패키지에서 @Transactional 어노테이션 선언 금지!
- {RULE] 단순 CRUD인 경우 @Transactional 어노테이션을 선언하지 않아도 된다.

5.4. Jasypt (yaml 암호화)

- yaml Jasypt 암호화 적용
  - 키(자바 argment로 받음)
    - -Djasypt.key=paletteHello
    - 개발 단계와 보안 적용 여부에 따라 적절하게 수정해서 사용해야 함
    - OS 환경변수 활용 가능함
  - 설정
    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml

```
spring.config.activate:
  on-profile: "common"

jasypt.encryptor.bean: jasyptStringEncryptor
```

    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/security/datasource/JasyptConfig.java

```
/**
  * 데이터소스 암복호화
  * @author leeiy
  */
  @Validated
  @Configuration
  public class JasyptConfig
  {
      @NotBlank
      @Value("${jasypt.key}")
      private String jasyptKey;
      /**
      * export JASYPT_KEY =palettehello
System.getenv("JASYPT_KEY");
      * or
      * -Djasypt.key=paletteHello
      * @return
      */
      @Bean("jasyptStringEncryptor")
      public StringEncryptor stringEncryptor()
      {
          PooledPBEStringEncryptor encryptor = new
PooledPBEStringEncryptor();
          SimpleStringPBEConfig config = new
SimpleStringPBEConfig();
          config.setPassword(jasyptKey);  //키는 시스템 환경변수 또
는 자바 -D 사용
          config.setAlgorithm("PBEWITHHMACSHA512ANDAES_256");
          config.setKeyObtentionIterations("1000");
```

```
            config.setPoolSize("1");
            config.setProviderName("SunJCE");

config.setSaltGeneratorClassName("org.jasypt.salt.RandomSaltGenera
tor");

config.setIvGeneratorClassName("org.jasypt.iv.RandomIvGenerator");
            config.setStringOutputType("base64");
            encryptor.setConfig(config);
            return encryptor;
        }
    }
```

- 기동 시 암/복호화 확인 가능
  - /palette-hkcloud-rnd-webm-run/src/main/java/kr/co/hkcloud/palette/PaletteWebmApplication.java

```
public class PaletteWebmApplication implements CommandLineRunner
{
    ...
    /**
     * 상용에서는 제거하고 빌드 필요함
     */
    public void run(String... args)
}
```

    - 기본값은 local 프로필에서만 동작하도록 설정되어 있음

5.5. Validation

- Validator 확장하여 사용 시
  - Validator 자바 생성
    - {RULE} util 패키지에 만든다
    - {RULE} 클래스명은 *Validator로 끝난다.
    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/board/util/BoardValidator.java

```
  @Component
  public class BoardValidator implements Validator
  {
      @Override
      public boolean supports(Class<?> clazz)
      {
          return false;
      }

      @Override
      public void validate(Object target, Errors errors)
      {
          TelewebJSON objJsonParams = (TelewebJSON) target;


if(objJsonParams.getHeaderString("TYPE").toString().equals("BIZ_SE
RVICE")) {

if(objJsonParams.getHeaderString("METHOD").toString().equals("list
")) {
                  selectValidate(target, errors);
              }
              else
if(objJsonParams.getHeaderString("METHOD").toString().equals("dele
te")) {
                  deleteValidate(target, errors);
              }
              else {
                  Object[] arg = new Object[1];
                  arg[0] = "METHOD";

errors.reject(PaletteValidationCode.UNDEFINED_METHOD.getCode(),
arg, PaletteValidationCode.UNDEFINED_METHOD.getMessage());
              }
          }
          else {
              Object[] arg = new Object[1];
              arg[0] = "BIZ_SERVICE";

errors.reject(PaletteValidationCode.UNDEFINED_TYPE.getCode(), arg,
PaletteValidationCode.UNDEFINED_TYPE.getMessage());
          }
```

```
        }
        ...
    }
```

- 컨트롤러에 주입하고 함수에서 사용
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/board/api/BoardRestController.java

```
    @RequiredArgsConstructor
    @RestController
    @Api(value = "BoardRestController",
        description = "실시간공지사항 REST 컨트롤러")
    public class BoardRestController
    {
        ...
        private final BoardValidator boardValidator;  //{1}

        ...

        /**
        * 게시판-목록-조회
        * @param  mjsonParams
        * @param  result
        * @return           TelewebJSON 형식의 조회결과 데이터
        */
        @ApiOperation(value = "게시판-목록-조회",
                      notes = "게시판 목록을 조회한다")
        @PostMapping("/api/board/list")
        public Object selectRtn(@TelewebJsonParam TelewebJSON
mjsonParams, BindingResult result) throws TelewebApiException
        {
            ...

            //DAO검색 메서드 호출
            String brdId = mjsonParams.getString("BRD_ID");

            if("1".equals(brdId) || "7".equals(brdId)) { // 시스템 공
지 사항 또는 챗봇 공지사항

                //Validation 체크
                boardValidator.validate(mjsonParams, result);  //{2}
                if(result.hasErrors()) { throw new
TelewebValidationException(result.getAllErrors()); } //{3}

                objRetParams =
boardService.selectRtnBrdSystem(mjsonParams);
            }
            ...

            objRetData1 =
boardService.selectRtnBrdCheck(mjsonParams);
```

```
            ...

            //최종결과값 반환
            return objRetParams;
        }
        ...
    }
```

- {1} 해당 Validator를 주입한다.
- {2} 함수를 호출한다.
- {3} 오류가 있는 경우 BindubgResult로 FE에 전달한다.

- 도메인(모델) 클래스를 사용 시
  - {RULE} 클래스는 domain 패키지 하위에 생성한다.
  - {RULE} 요청과 응답 클래스를 만든다.
  - {RULE} 요청은 클래스명이 *Request로 끝난다.
  - {RULE} 응답은 클래스명이 *Response로 끝난다.
  - {RULE} 클래스가 많아지는 걸 지양하기 위해 내부 Static Inner Class를 활용한다.
  - @valid vs @validated
    - 참고 URL : @Valid, @Validated를 이용한 데이터 유효성 검증
  - 도메인(모델) 클래스 생성
    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/domain/FileRequest.java

```
/**
* 파일 요청 도메인
* @author leeiy
*/
public class FileRequest
{
    //서비스단(@Validated groups용으로 Request단에만 존재시킨다)
    public interface GroupService{};

    /**
    * 파일 업로드 단 건 요청
    * @author leeiy
    */
    @NoArgsConstructor
    @AllArgsConstructor
    @Getter
    @Setter
    @Builder
    @ToString(exclude = "file")
    public static final class FileUploadRequest
    {
        @NotNull
        private MultipartFile file;

        @FileGroupKeyConstraint(groups =
```

```
{FileRequest.GroupService.class})
          private String fileGroupKey; // 파일그룹키

          @NotNull(groups = {FileRequest.GroupService.class})
          private FileAccessType fileAccessType;  //파일엑세스유형

          @Setter(AccessLevel.NONE)  //수정하지 말 것! 제한적으로
builder()로만 엑세스 하세요.
          @Builder.Default
          private String aspCustKey = ((PaletteUserDetailsVO)
(SecurityContextHolder.getContext().getAuthentication().getPrincip
al())).getAspCustKey();

          @Setter(AccessLevel.NONE)  //수정하지 말 것! 제한적으로
builder()로만 엑세스 하세요.
          @Builder.Default
          private String procId = ((PaletteUserDetailsVO)
(SecurityContextHolder.getContext().getAuthentication().getPrincip
al())).getUserId();
      }
      ...
  }
```

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/domain/FileResponse.java

```
  /**
  * 파일 응답 도메인
  * @author leeiy
  */
  public class FileResponse
  {
      /**
      * 파일 업로드 응답
      * @author leeiy
      */
      @Getter
      @Builder
      public static final class FileUploadResponse
      {
          @NotBlank
          @FileGroupKeyConstraint
          private String fileGroupKey; // 파일그룹키

          @NotBlank
          @FileGroupKeyConstraint
          private String fileKey; // 파일키

          @NotBlank
          private String originalFilename; // 파일명
```

```
            @PositiveOrZero
            private long fileSize; // 크기

            @NotBlank
            private String fileSizeDisplay; //크기(KB)

            @PositiveOrZero
            private int dnlodCnt; // 다운로드수

            @NotBlank
            private String fileExts; // 확장자
        }
        ...
    }
```

- 컨트롤러에 어노테이션 선언
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/api/FileUploadRestController.java

```
public class FileUploadRestController
{
    private final FileUploadValidator fileUploadValidator;
    ...

    @PostMapping("/api/file/{busiType}/{pathType}/uploads")
    public ResponseEntity<PaletteApiResponse<?>>
uploadFiles(@FileRuleProperties @Valid final
FilePropertiesResponse filePropertiesResponse, @Valid final
FileUploadRequests fileUploadRequests, final BindingResult
bindingResult) throws PaletteApiException
    {
        ...
        //유효성 체크
        fileUploadValidator.validate(validateMap, bindingResult);
        if(bindingResult.hasErrors()) {
            for(ObjectError error : bindingResult.getAllErrors())
{
                log.error("error.toString : {}",
error.toString());
            }
            throw new
PaletteValidationException(bindingResult.getAllErrors());
        }
        ...
    }
    ...
}
```

- **TelewebJSON을 사용하려는 경우 사용할 수 없음!**

5.6. Exception

- @RestControllerAdvice
  - @Order 적용
- 텔레웹 익셉션

```
class TelewebSecurityException extends RuntimeException {}
class TelewebValidationException extends RuntimeException {}
class TelewebException extends RuntimeException {}
class TelewebWebException extends TelewebException {}
class TelewebApiException extends TelewebException {}
class TelewebAppException extends TelewebException {}
class TelewebDaoException extends TelewebException {}
class TelewebUtilException extends TelewebException {}
class TelewebMainParamSignatureException extends RuntimeException {}
```

- 팔레트 익셉션

```
class PaletteSecurityException extends RuntimeException {}
class PaletteValidationException extends RuntimeException {}
class PaletteException extends RuntimeException {}
class PaletteWebException extends PaletteException {}
class PaletteApiException extends PaletteException {}
class PaletteAppException extends PaletteException {}
class PaletteDaoException extends PaletteException {}
class PaletteUtilException extends PaletteException {}
```

5.7. RestTemplate (대외 Restful API)

- 설정
  - 참고 URL: RestTemplate 설정 변경하기
  - /palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/connection/RestTemplateConfig.java

```
/**
  * RestTemplate 설정
  * @author leeiy
  */
  @Slf4j
  @RequiredArgsConstructor
  @Configuration
  public class RestTemplateConfig
  {
      private static final int MAX_CONN_TOTAL     = 200;    //ea
      private static final int MAX_CONN_PER_ROUTE = 20;     //ea
      private static final int CONNECT_TIMEOUT    = 5000;   //ms
      private static final int READ_TIMEOUT       = 30000;  //ms

      private final ProxyProperties proxyProperties;


      @Bean
      public RestTemplate restTemplate()
      {
          log.info("Loading RestTemplate!!");
          HttpComponentsClientHttpRequestFactory requestFactory = new
HttpComponentsClientHttpRequestFactory();

          // @formatter:off
          HttpClient client;
          if(proxyProperties.isEnabled()) {
              String proxyDomain = proxyProperties.getDomain();
              int proxyPort = proxyProperties.getPort();
              String proxySchema = proxyProperties.getSchema();
              HttpHost proxy = new HttpHost(proxyDomain, proxyPort,
proxySchema);
              log.debug("PROXY INFO:\n\n\tHOST: {}\n\tPORT:
{}\n\tSCHEMA: {}\n\n", proxy.getHostName()

, proxy.getPort()

, proxy.getSchemeName());

              client = HttpClientBuilder.create()
                                        .setMaxConnTotal(MAX_CONN_TOTAL)

.setMaxConnPerRoute(MAX_CONN_PER_ROUTE)
                                        .setRoutePlanner(new
```

```
DefaultProxyRoutePlanner(proxy) {
                                                public HttpHost
determineProxy(HttpHost target, org.apache.http.HttpRequest request,
HttpContext context) throws HttpException {
                                                    return
super.determineProxy(target, request, context);
                                                }
                                            })
                                            .build();
            }
            else
            {
                client = HttpClientBuilder.create()
                                            .setMaxConnTotal(MAX_CONN_TOTAL)

.setMaxConnPerRoute(MAX_CONN_PER_ROUTE)
                                            .build();
            }
            // @formatter:on

            requestFactory.setHttpClient(client);
            requestFactory.setConnectTimeout(CONNECT_TIMEOUT);
            requestFactory.setReadTimeout(READ_TIMEOUT);

            //로그가 필요없는 경우
  //          return new RestTemplate(factory);

            //로그가 필요한 경우
            //execute() 실행 시 Stream이 닫힘
            RestTemplate restTemplate = new RestTemplate(new
BufferingClientHttpRequestFactory(requestFactory));
            restTemplate.setInterceptors(Collections.singletonList(new
RequestResponseLoggingInterceptor()));

            return restTemplate;
        }
    }


    @Slf4j
    class RequestResponseLoggingInterceptor implements
ClientHttpRequestInterceptor
    {
        @Override
        public ClientHttpResponse intercept(HttpRequest request, byte[]
body, ClientHttpRequestExecution execution) throws IOException
        {
            HttpHeaders rqHeader = request.getHeaders();
            URI rqURI = request.getURI();
            HttpMethod rqMethod = request.getMethod();
            String rqMethodValue = request.getMethodValue();
            String rqbody;
            if(body != null) {
                rqbody = (body.length > 2000) ? "[...Big body...]" :
```

```
body.toString();
            }
            else {
                rqbody = "body null~!!";
            }

            // @formatter:off
            log.debug(new StringBuffer().append("\n>>>>>REQUEST INFO\n")
                                        .append("HEADER
:").append(rqHeader.toString()).append("\n")
                                        .append("URI
:").append(rqURI.toString()).append("\n")
                                        .append("METHOD
:").append(rqMethod.toString()).append("\n")

.append("METHOD_VALUE:").append(rqMethodValue).append("\n")
                                        .append("BODY
:").append(rqbody).append("\n\n")
                                        .toString());

            ClientHttpResponse response = execution.execute(request,
body);

            HttpHeaders rpHeader = response.getHeaders();
            HttpStatus rpStatusCode = response.getStatusCode();
            String rpStatusText = response.getStatusText();
            String rpbody = StreamUtils.copyToString(response.getBody(),
StandardCharsets.UTF_8);

            log.debug(new StringBuffer().append("\n>>>>>RESPONSE INFO\n")
                                        .append("HEADER
:").append(rpHeader.toString()).append("\n")
                                        .append("STATUS_CODE
:").append(rpStatusCode.toString()).append("\n")
                                        .append("STATUS_TEXT
:").append(rpStatusText).append("\n")
                                        .append("BODY
:").append(rpbody.length() > 2000 ? "[...Big body...]" :
rpbody).append("\n\n")
                                        .toString());
            // @formatter:on
            return response;
        }
    }
```

- 사용법
  - 컴포넌트 주입

```
private final RestTemplate restTemplate;
```

- 해당 함수에서 사용

```
restTemplate.postForObject(URI.create(Uri), request, String.class);
```

- 참고 - 전화-CTI 연동
  - /palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/infra/tplex/cti/util/IvrRestTemplateUtils.java

```
/**
* http 통신을 위한 RestTemplate 유틸 TODO 사용하는데 확인 필요
*
* @author R&D
*/
@Slf4j
@RequiredArgsConstructor
@Component
public class IvrRestTemplateUtils
{
    private final RestTemplate      restTemplate;
    private final PaletteProperties paletteProperties;

    public String sendRestTemplate(MultiValueMap<String, String>
parameters)
    {
        // http 통신 결과가 들어오는 문자열 변수
        String result = null;

        // http 통신의 대상이 되는 URL (CTI서버)
        // Properties 가져오는 것으로 변경
        String Uri = paletteProperties.getCtiServer().getUrl() + ":" +
paletteProperties.getCtiServer().getRecordingPort() + "/API/";
        // http 통신의 헤더값
        HttpHeaders headers = new HttpHeaders();
        headers.add("User-Agent", "Mozilla/5.0");
        headers.add("Accept-Charset",
StandardCharsets.UTF_8.toString());

        // http 통신의 value값
        HttpEntity<MultiValueMap<String, String>> request = new
HttpEntity<>(parameters, headers);

        //POST 방식으로 cti 서버로 리퀘스트를 전송한다. 인자 (URL,
HttpEntity request, 받을 클래스 형식);
        result = restTemplate.postForObject(URI.create(Uri), request,
String.class);

        return result;
    }
}
```

5.8. Swagger2

5.9. Grid 관련

5.10. 파일 관련

- [RULE] 모듈별로 저장소를 분리한다.
- {RULE} 파일은 모두 PLT_FILE DB 테이블에서 관리한다.
- {RULE} 파일 다운로드는 FILE_GROUP_KEY와 FILE_KEY로 핸들링한다.
- {RULE} 대외 파일다운로드는 ~/infra/file 패키지 구조를 따른다.
- {RULE} FE에서는 api로 진입하고, BE에서는 util로 호출한다.
- 기본적으로 File과 Db(Blob) 저장을 지원한다.
- 구조
    - 파일 업로드
        - dropzone js lib 사용됨
            - 공식 사이트 : dropzone js
        - 설정
            - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml

```
#########################################################
#
# 파일(file) 속성
# -------------------------------------------------
# include 'file' profile
#########################################################
---
spring.config.activate:
  on-profile: "file"
spring.servlet:
# ----------------------------------------
# MULTIPART (MultipartProperties)
# ----------------------------------------
  multipart:
    # Whether to enable support of multipart uploads.
    enabled: true
    # Threshold after which files are written to disk.
    file-size-threshold: 100KB                          #{1}
    # Intermediate location of uploaded files.
    location: D:/PALETTE/hkcloud/rnd/repositorytemp      #{2}
    # Max file size.
    max-file-size: 5MB                                  #{3}
    # Max request size.
    max-request-size: 20MB                              #{4}
    # Whether to resolve the multipart request lazily at the
```

```
time of file or parameter access.
    #resolve-lazily: false
```

- {1},{3],{4}의 크기는 고객사 환경에 따라 적절하게 설정이 필요할 수 있음
- {2} 멀티파트 파일 임시 파일 저장소 경로

- ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml

```
# ---------------------------------------
# FileProperties
# ---------------------------------------
file:
  enabled: true                                    #{1}
  file-upload-lib: dropzone                        #{2}
  repository:
    # ---------------------------------------
    # 루트 저장소
    # ---------------------------------------
    root-dir: file:///d:/PALETTE/hkcloud/rnd/repository/web
#{3}

    # ---------------------------------------
    # 팔레트 저장소
    # target-type: FILE / DB (저장소 대상 타입)
    # ---------------------------------------
    palette:                                       #{4}
      files:                                       #{5}
        target-type: FILE                          #{6}
        enabled: true                              #{7}
        max-files: 10                              #{8}
        max-filesize: 2MB                          #{9}
        accepted-files:                            #{10}
          - "pdf"
          - "xlxs"
        dir: ${file.repository.root-dir}/palette/file  #{11}
        temp-dir: ${file.repository.root-
dir}/palette/file/temp
    ...
    # ---------------------------------------
    # 채팅 저장소
    # ---------------------------------------
    chat:
      ...
      files:                                       #{12}
        target-type: FILE
        enabled: true
        max-files: 10
        max-filesize: 2MB
        accepted-files:
          - "pdf"
        dir: ${file.repository.root-dir}/chat/file
        temp-dir: ${file.repository.root-dir}/chat/file/temp
```

```
      images:                                                #{13}
        target-type: FILE
        enabled: true
        max-files: 10
        max-filesize: 2MB
        accepted-files:
          - "png"
          - "jpg"
          - "gif"
        dir: ${file.repository.root-dir}/chat/images
        temp-dir: ${file.repository.root-
dir}/chat/images/temp
    ...
```

- {1} 파일 활성 여부
- {2} 업로드 라이브러리 이름
- {3} 저장소 루트 경로
- {4} 팔레트 저장소
- {5} 경로 유형 (file,image 등)
- {6} 저장소 대상 유형 (FILE/DB)
- {7} 팔레트 저장소 활성 여부
- {8} 동시에 첨부 가능한 파일 갯수
- {9} 개 당 파일 제한 크기
- {10} 허용된 확장자들
- {11} 저장 경로
- {12} 채팅-파일 저장소를 의미함
- {13} 채팅-이미지 저장소를 의미함
  - 이미지의 경우 확장자가 제공되며, 다운로드 시 스트림으로 다운로드 받아 바로 표시할 수 있음

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/properties/file/FileProperties.java

```
/**
* 파일(file) 속성
* @author RND
*/
@Getter
@RequiredArgsConstructor
@Validated
@ConstructorBinding
@ConfigurationProperties(prefix = "file")
public class FileProperties
{
    @NotNull
    private final boolean enabled;

    @NotNull
    private final FileUploadLib fileUploadLib;
```

```
    private final Repository repository;


    @Getter
    @RequiredArgsConstructor
    public static final class Repository
    {
        @NotNull
        private final File rootDir;

        private final Palette palette;
        ...

        @Getter
        @RequiredArgsConstructor
        public static final class Palette
        {
            private final Files files;


            @Getter
            @RequiredArgsConstructor
            public static final class Files
            {
                @NotNull
                private final RepositoryTargetType
targetType;
                @NotNull
                private final boolean              enabled;
                @NotNull
                private final int                  maxFiles;
                @DataSizeUnit(DataUnit.MEGABYTES)
                @NotNull
                private final DataSize
maxFilesize;
                @NotNull
                private final List<String>
acceptedFiles;
                @NotNull
                private final Path                 dir;
                @NotNull
                private final Path                 tempDir;
            }
            ...
        }
        ...
    }
}
```

- JAVA package

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/** 하위 패키지
- FE에서 api로 진입
  - 흐름
    - Method Argument Resolver -> 컨트롤러
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/args/FileRulePropertiesArgumentResolver.java

```
  /**
  * 파일 경로구분에 따른 프로퍼티
  * @author RND
  */
  @Slf4j
  @RequiredArgsConstructor
  @Component
  public class FileRulePropertiesArgumentResolver
implements HandlerMethodArgumentResolver
  {
      private final FileRulePropertiesUtils
fileRulePropertiesUtils;

      /**
      * Method parameter에 대한 Argument Resovler로직 처
리 ㄴ 파일 경로구분에 따른 프로퍼티를 함수의 인수에 자동으로
설정해 줌
      */
      @Override
      public Object resolveArgument(MethodParameter
parameter, ModelAndViewContainer mavContainer,
NativeWebRequest webRequest, WebDataBinderFactory
binderFactory)
      {
          @SuppressWarnings("unchecked")
          Map<String, Object> uriTemplateVars =
(Map<String, Object>)
webRequest.getAttribute(HandlerMapping.URI_TEMPLATE_VARI
ABLES_ATTRIBUTE, RequestAttributes.SCOPE_REQUEST);
          final RepositoryBusiType busiType =
RepositoryBusiType.valueOf(uriTemplateVars.get("busiType
").toString());
          final RepositoryPathType pathType =
RepositoryPathType.valueOf(uriTemplateVars.get("pathType
").toString());
          FilePropertiesResponse filePropertiesResponse
= fileRulePropertiesUtils.getProperties(busiType,
pathType);
          return filePropertiesResponse;
      }
      @Override
      public boolean supportsParameter(MethodParameter
```

```
parameter)
      {
          //@FileRuleProperties 어노테이션이 붙은 파라미터
에 대해 적용
          //ㄴ Controller에서만 사용해야 함
          return
parameter.hasParameterAnnotation(FileRuleProperties.clas
s);

      }
  }
```

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/api/FileUploadRestController.java

```
  @Slf4j
  @RestController
  public class FileUploadRestController
  {
      ...


@PostMapping("/api/file/{busiType}/{pathType}/uploads")
//{1}
      public ResponseEntity<PaletteApiResponse<?>>
uploadFiles(@FileRuleProperties @Valid final
FilePropertiesResponse filePropertiesResponse, @Valid
final FileUploadRequests fileUploadRequests, final
BindingResult bindingResult) throws PaletteApiException
//{2}
      {
          ...

          //파일 저장
          List<FileUploadResponse> uploadResponseList =
fileUploadUtils.store(filePropertiesResponse,
fileUploadRequests);

          //응답
          return new ResponseEntity<PaletteApiResponse<?
>>(PaletteApiResponse.res(uploadResponseList),
HttpStatus.OK);
      }
  }
```

- BE에서 util로 아래 함수를 호출하여 처리한다.
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/util/FileUploadUtils.java

```
    @RequiredArgsConstructor
    @Validated
    @Component
    public class FileUploadUtils
    {
        ...
        @NotNull
        public List<FileUploadResponse> store(@Valid final
  FilePropertiesResponse filePropertiesResponse, @Valid
  final FileUploadRequests fileUploadRequests) throws
  PaletteAppException
        ...
    }
```

- ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/util/FileUploadUtils.java

```
@Slf4j
@RequiredArgsConstructor
@Validated
@Component
public class FileUploadUtils
{
    private final FileRuleUtils      fileRuleUtils;
    private final FileStorageService fileStorageService;
    private final FileStorageService fileStorageDbService;


    /**
    * 파일 저장
    *
    * @param  filePropertiesResponse
    * @param  fileUploadRequests
    * @return
    * @throws PaletteAppException
    */
    @NotNull
    public List<FileUploadResponse> store(@Valid final
FilePropertiesResponse filePropertiesResponse, @Valid final
FileUploadRequests fileUploadRequests) throws
PaletteAppException
    {
        //그룹키 생성 (있으면 사용하고, 없으면 생성한다)
        String fileGroupKey =
fileRuleUtils.ifCreatFileGroupKey(fileUploadRequests);

        //응답용
        List<FileUploadResponse> uploadResponseList = new
ArrayList<>();

        //파일처리
```

```
        MultipartFile[] files =
fileUploadRequests.getUserfiles();
        Arrays.asList(files).stream().forEach(file -> {
            //파일 empty 체크
            if(file.isEmpty()) {
                //무시한다.
                log.info("{} file empty skip!",
file.getOriginalFilename());
                return; // only skips this iteration.
            }
            ...

            //저장소대상구분
            final RepositoryTargetType targetType =
filePropertiesResponse.getTargetType();
            switch(targetType)
            {
                //파일 / 에디터
                case FILE:
                case EDITOR:
                {

uploadResponseList.add(fileStorageService.store(fileProperties
sResponse, fileUploadRequest));
                    break;
                }
                //DB
                case DB:
                {

uploadResponseList.add(fileStorageDbService.store(fileProperti
esResponse, fileUploadRequest));
                    break;
                }
            }
        });
        return uploadResponseList;
    }
}
```

- BE에서 공개 영역으로 저장하려는 경우
  - ~/palette-hkcloud-rnd-webm-chat/src/main/java/kr/co/hkcloud/palette/core/chat/messenger/hkcdv/kakaobzc/app/HkcdvKakaobzcReceiveMessageImpl.java

```
    ...
    final RepositoryBusiType busiType =
  RepositoryBusiType.chat;    //채팅
    final RepositoryPathType pathType =
  RepositoryPathType.images;  //이미지(고객)
    final FilePropertiesResponse fileProperties =
```

```
fileRulePropertiesUtils.getProperties(busiType,
pathType);
  //이미지 레파지토리 저장
  final JSONObject jsonFile =
teletalkReceiveUtils.savePhototoRepository(messageJson,
objParams, callTypCd, fileProperties);
  ...
```

- ~/palette-hkcloud-rnd-webm-chat/src/main/java/kr/co/hkcloud/palette/core/chat/messenger/util/TeletalkReceiveUtils.java

```
  /**
  * 전송 받은 이미지 저장 ( 서버 repository )
  */
  public JSONObject savePhototoRepository(JSONObject
rcvJson, TelewebJSON objParams, String callTypCd,
FilePropertiesResponse fileProperties) throws
TelewebUtilException
  {
      ...
      //URI 생성
      URI imageUri = URI.create(imageTalkUrl);2

      //임시파일 생성
      File tempFile = restTemplate.execute(imageUri,
HttpMethod.GET, null, clientHttpResponse -> {
          File ret =
File.createTempFile("image_download", "tmp");
          StreamUtils.copy(clientHttpResponse.getBody(),
new FileOutputStream(ret));
          return ret;
      });
      ...

      // @formatter:off
      //파일 to 멀티파트 변환
      final MultipartFile file =
fileConvertUtils.toMultipartFile(tempFile);

      switch(fileProperties.getTargetType())
      {
          default:
          {
              //어레이로 변환
              final MultipartFile[] files = {file};

              //그룹키 생성 (있으면 사용하고, 없으면 생성한
다)
              String fileGroupKey =
fileRuleUtils.ifCreatFileGroupKey(new
```

```
 FileUploadRequests());
                FileUploadRequests fileUploadRequests =
 FileUploadRequests.builder()

 .userfiles(files)

 .aspCustKey(rcvJson.getString("ASP_CUST_KEY")) //{1}

 .fileGroupKey(fileGroupKey)

 .procId("system")  //{2}

 .build();

                //파일 저장
                List<FileUploadResponse>
 fileUploadResponseList =
 fileUploadUtils.store(fileProperties,
 fileUploadRequests);
                ...
                break;
            }
        }
        // @formatter:on
        ...
        return retJson;

    }
```

  - {1} 고객사 키를 전달한다.
  - {2} 처리자 ID를 전달한다.
- 공개영역 다운로드 하려면 아래 **'파일 다운로드 > 공개(Infra/DMZ) 영역에서의 다운로드'** 참고...

- HTML
  - ~/palette-hkcloud-rnd-webm-common/src/main/resources/templates/fragments/file/file-headerinc.html

```
    <!--/* 첨부파일 기본 설정 (script 위치 변경하면 안됨) */-->
    <script th:inline="javascript">
    var FileBaseConfig = {
    fileUploadLib: function() {
        var libname = /*[[ ${fileProperties.fileUploadLib}
]]*/ null;
        //console.log("fileUploadLib:", libname);
        return libname;
    },
    uploadUri: function() {
        var uri = /*[[ ${fileProperties.uploadUri} ]]*/ null;
        //console.log("uploadUri:", uri);
        return uri;
    },
```

```
    downloadUri: function() {
        var uri = /*[[ ${fileProperties.downloadUri} ]]*/
null;
        //console.log("downloadUri:", uri);
        return uri;
    },
    params: {},
    nowFiles: 0,
    maxFiles: function() {
        var maxFiles = /*[[ ${fileProperties.maxFiles} ]]*/
null;
        //console.log("maxFiles:", maxFiles);
        return maxFiles;
    },
    maxFilesize: function() {
        var maxFilesize = /*[[
${fileProperties.getMaxFilesize().toMegabytes()} ]]*/ null;
        //console.log("maxFilesize:", maxFilesize);
        return maxFilesize;
    },
    acceptedFiles: function() {
        var arrys = /*[[ ${fileProperties.acceptedFiles} ]]*/
null;
        if(!arrys) return null;
        var str = [];
        for(var i=0; i < arrys.length; i++)
        {
            //console.log(i+"===>"+arrys[i]);
            str[i] = "."+arrys[i];
        }
        return str.join(",");
    },
    // Headers (they are sent all at once)
    requestHeaders: {},
    requestHeadersNames: {},
    setRequestHeader: function(name, value) {
        name = this.requestHeadersNames[ name.toLowerCase() ]
=
                this.requestHeadersNames[ name.toLowerCase()
] || name;
        this.requestHeaders[ name ] = value;
        return this.requestHeaders;
    },
    fileBusiType:function(){
        var busiType = /*[[ ${fileProperties.busiType} ]]*/
null;
        return busiType;
    },
    };
    </script>
    <!--/* palette-core, palette-validations, palette-utils
공통화 필요함 - 20210622 */-->
    <script type="text/javascript" th:src="@{/core/palette-
core-1.0.0.js}"></script>
```

```
    <script type="text/javascript" th:src="@{/core/palette-
validations-1.0.0.js}"></script>
    <script type="text/javascript"
th:src="@{/core/util/palette-utils-1.0.0.js}"></script>

    <script type="text/javascript" th:src="@{/file/file-
atchmnfl-dialog-1.0.0.js}"></script>
    <script type="text/javascript" th:src="@{/file/file-
atchmnfl-grid-1.0.0.js}"></script>

    <script type="text/javascript"
th:if="${#strings.equals(fileProperties.fileUploadLib,
'dropzone')}" th:src="@{/resources/js/dropzone-
5.9.2/dist/min/dropzone.min.js?v=1}"></script>
    <script type="text/javascript"
th:if="${#strings.equals(fileProperties.fileUploadLib,
'dropzone')}" th:src="@{/file/file-atchmnfl-dropzone-
1.0.0.js}"></script>

    <script type="text/javascript" th:src="@{/file/file-
atchmnfl-download-1.0.0.js}"></script>
    <script type="text/javascript" th:src="@{/file/file-
atchmnfl-1.0.0.js}"></script>
```

- yml 속성을 js로 내려주고, 사용할 자바스크립트 파일을 결정한다.

- ~/palette-hkcloud-rnd-webm-common/src/main/resources/templates/fragments/file/file-bodyinc-grid.html

```
   <!-- 첨부파일 그리드 시작 -->
   <div class="tt-mt-15" id="divFileArea">
        <div class="tt-form-wrap">
            <div class="twb-ui-box-head">
                <h1>파일목록</h1>
                <div class="actions">
                    <button th:if="${fileProperties.enabled
== true}" type="button" class="tt-btn-free" id="btnAtchmnfl">
파일등록</button>
                    <button type="button" class="tt-btn-free
is-fill" id="btnDeleteRow">파일삭제</button>
                </div>
            </div>
            <div class="tt-grid-frame">
                <div id="divFileListGrid" gridTitle="파일목
록"></div>
            </div>
        </div>
   </div>
   <!-- 첨부파일 그리드 끝 -->
```

- 해당 파일 처리 업무에 공통 첨부파일 그리드 영역을 만들어 준다.

- ~/palette-hkcloud-rnd-webm-common/src/main/resources/templates/fragments/file/file-bodyinc-dialog.html

```
<!-- 첨부파일 다이얼로그 시작 -->
<div id="divFileDialog" style="display:none;">
    <div class="tt-window-body">
        <span class="tt-info-txt is-fill tt-mb-5">확장자는 <em
th:each="item : ${fileProperties.getAcceptedFiles()}"
th:text="(${itemStat.last})?${item}:${item}+', '"></em>
        , 용량은 <em
th:text="${@fileRulePropertiesUtils.toFileSizeDisplay(filePro
perties.getMaxFilesize().toBytes())} + ' 이하'"></em>이며, 최
대 <em th:text="${fileProperties.maxFiles} + '개'"></em>까지
등록 가능합니다.</span>
        <div id="atchmnflForm" class="palette-dropzone"
style="height: 220px;">
            <div class="dz-message" data-dz-message>
                <i class="tt-icon-plus-lg-bold"></i>
                <span class="dz-message-txt">여기로 업로드 할
파일을 드래그하거나, 클릭하세요.</span>
            </div>
        </div>
    </div>
</div>
<!-- 첨부파일 다이얼로그 끝 -->
```

  - 그리드의 **파일추가** 버튼을 클릭할 때 뜨는 공통 파일 처리 다이얼로그창이다.
  - 허용된 확장자, 용량, 최대 파일 갯수 정보를 제공한다.

- JS
  - ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/file/file-atchmnfl-1.0.0.js

```
var FileAtchmnfl = {

/**************************************************************
*******************
    * Init Function : 초기 로드 시점 정의
    * - 실행 시점 문제가 있으므로, 파일을 사용하려는 페이지에서
컨트롤 함
    * - 먼저 실행되어야 함

**************************************************************
******************/
    init: function() {
        gFileUploadLibName = FileBaseConfig.fileUploadLib();
        if(gFileUploadLibName && gFileUploadLibName ==
"dropzone") {
            //Forbid the automatic search of all elements:
            Dropzone.autoDiscover = false;
        }
```

```
        ...
    },
    ...
    //-----------------------------------------------------------
------------------------
    // GRID: 그리드 정의
    //-----------------------------------------------------------
------------------------
    defineGrid: function() {
        if(!FileAtchmnflGrid.nowCssSelector) {

FileAtchmnflGrid.defineGrid("div#divFileListGrid");
        }
    },
    //-----------------------------------------------------------
------------------------
    // 업로드 라이브러리 정의
    //-----------------------------------------------------------
------------------------
    defineUploadLib: function() {
        //Dropzone 설정
        if(gFileUploadLibName && gFileUploadLibName ==
"dropzone") {

FileAtchmnflDropzone.defineDropzone("div#atchmnflForm");
        }
    },
    //-----------------------------------------------------------
------------------------
    // 다이얼로그 정의
    //-----------------------------------------------------------
------------------------
    defineDialog: function() {
        if ($("div#divFileDialog").length) {

FileAtchmnflDialog.defineDialog("div#divFileDialog");
        }
    },
    ...

/************************************************************
********************
     * Event Object : 화면에 디자인 된 버튼 및 오브젝트 이벤트와
호출할 함수를 정의

************************************************************
********************/
    defineEvent: function() {
        jQuery("button#btnAtchmnfl").click(function() {
FileAtchmnflDialog.getDialog().openDhxWindow(); }); //파일 등
록 버튼
        jQuery("button#btnDeleteRow").click(function() {
FileAtchmnfl.deleteFiles(); }); //파일 삭제 버튼
    },
```

```
    ...
}
```

- ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/file/file-atchmnfl-grid-1.0.0.js

```
var FileAtchmnflGrid = {
    ...
    defineGrid: function(customCssSelector) {
        ...
        //첨부파일목록 그리드
        $(FileAtchmnflGrid.nowCssSelector).dhxGrid({
            height          : 140,              //(필수){int} 그리드 높이
            ...
        });
    }
    ...
}
```

- ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/file/file-atchmnfl-dialog-1.0.0.js

```
var FileAtchmnflDialog = {
    ...
    defineDialog: function(customCssSelector) {
        ...
        $( FileAtchmnflDialog.nowCssSelector ).dhxWindow({
            text:"파일등록",
            ...
        });
        ..
    }
    ...
}
```

- ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/file/file-atchmnfl-dropzone-1.0.0.js

```
...
var FileAtchmnflDropzone = {
   ...
    defineDropzone: function(customCssSelector) {
        ...
        $( FileAtchmnflDropzone.nowCssSelector ).dropzone({
            headers: FileBaseConfig.requestHeaders,
            ...
```

```
            });
        }
        ...
    }
```

- 파일 다운로드
  - 내부 영역에서의 다운로드
    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/file/api/FileDownloadRestController.java

```
public class FileDownloadRestController
{
    ...
    @PostMapping("/api/file/{busiType}/{pathType}/download")
    public ResponseEntity<Resource>
downloadFile(@FileRuleProperties @Valid final
FilePropertiesResponse filePropertiesResponse
                                                , @Valid final
FileDownloadRequest fileDownloadoadRequest
                                                , final
HttpServletRequest httpServletRequest) throws
PaletteApiException
    {
        ...
    }
    ...
}
```

    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/file/file-atchmnfl-download-1.0.0.js

```
var FileAtchmnflDownload = {
/************************************************************
********************
* Init Function : 초기 로드 시점 정의
* - 실행 시점 문제가 있으므로, 파일을 사용하려는 페이지에서 컨트롤
함
* - 먼저 실행되어야 함
************************************************************
********************/
    init: function() {
        //페이지 초기화 처리
        FileAtchmnflDownload.initPage();
        //이벤트 정의
        FileAtchmnflDownload.defineEvent();
    },
/************************************************************
********************
* InitPage Function : 화면이 초기 로드 시점 정의
************************************************************
********************/
    ...
    //-------------------------------------------------------
------------------------
```

```
    // GRID: 그리드 정의
    //-------------------------------------------------------
-----------------------
    defineGrid: function() {
        if(!FileAtchmnflGrid.nowCssSelector) {

FileAtchmnflGrid.defineGrid("div#divFileListGrid");
        }
    },
    ...
/*************************************************************
*******************
* Main Functions: 주요 기능을 처리하는 함수 정의
*************************************************************
******************/
//-------------------------------------------------------------
-------------------
// 조회 함수 정의 :
//-------------------------------------------------------------
-------------------
    /**
    * 파일 목록 조회
    *
    * @param fileGroupKey 파일그룹키
    * @param isInitGrid   그리드초기화여부
    */
    selectFiles: function(fileGroupKey, isInitGrid) {

        ...

        var selectListResponse = PaletteCore.apiRequest({
            url: "/api/file/select-list",
            data: {
                fileGroupKey: fileGroupKey
            }
        });
        selectListResponse.done(function(data) {
            if(data.data && data.data.length > 0)
            {
                var objJsonReturn = new TelewebJson();
                objJsonReturn.setDataObject(data.data);
                grid.loadGridData(objJsonReturn);
            }
        });
    },

//-------------------------------------------------------------
-------------------
// 처리 함수 정의 :
//-------------------------------------------------------------
-------------------
    //다운로드 조회 수 증가 처리
    processDownloadCnt: function(downloadCntRequestData) {
        ...
```

```
        var processDownloadCntResponse =
PaletteCore.apiRequest({
            url: "/api/file/update-dnlod-cnt",
            data: downloadCntRequestData
        });
        processDownloadCntResponse.done(function() {
            FileAtchmnflDownload.selectFiles();  //재조회
        });
    },
//-------------------------------------------------------------
--------------------
// 파일 함수 정의 :
//-------------------------------------------------------------
--------------------
    /**
    * 파일 다운로드
    * @param (object) [필수] request.url           다운로드 주
소
    * @param (object) [필수] request.fileGroupKey  파일그룹키
    * @param (object) [필수] request.fileKey       파일키
    */
    downloadFile: function(downloadFileRequest) {
        ...
        var downloadFileResponse = jQuery.ajax({
            type: "POST",
            url: downloadFileRequest.url,
            cache: false,
            data: {
                fileGroupKey:
downloadFileRequest.fileGroupKey,
                fileKey: downloadFileRequest.fileKey
            },
            xhr: function () {
                ...
                xhr.onreadystatechange = function () {
                    if (xhr.readyState == 2) {
                        if (xhr.status == 200) {
                            xhr.responseType = "blob";
                        } else {
                            xhr.responseType = "text";
                        }
                    }
                };
                return xhr;
            },
            ...
        });
        downloadFileResponse.done(function(data, textStatus,
jqXHR) {
            ...
                //다운로드 수 증가
                var downloadCntRequestData = {
                    fileGroupKey:
downloadFileRequest.fileGroupKey,
```

```
                    fileKey: downloadFileRequest.fileKey
                };

FileAtchmnflDownload.processDownloadCnt(downloadCntRequestDat
a);
            ...
        });
        downloadFileResponse.fail(function(jqXHR, textStatus,
errorThrown) {
            ...
        });
    },
    ...
};
```

- download js 파일이 별도로 존재하는 이유는 권한에 따라 보기만 가능한 경우 때문이다.
- 보기 권한만 있는 경우 download js를 include하여 사용한다.

- 공개(Infra/DMZ) 영역에서의 다운로드
  - {RULE} 파일 저장 시 FileAccessType이 PUBLIC으로 저장된 파일만 다운로드할 수 있으며, 있어야 함.
  - {RULE} RepositoryPathType이 images인 경우만 공개되어야 함.
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/infra/file/api/InfraFileDownloadRestController.java

```
public class InfraFileDownloadRestController
{
    ...
    @GetMapping(value =
"/infra/api/file/{aspCustKey}/{busiType}/{pathType}/{fileGrou
pKey}/{fileKey}")
    public ResponseEntity<Resource>
downloadImageResourceInfra(@FileRuleProperties @Valid final
FilePropertiesResponse filePropertiesResponse
                                                        ,
@Valid final InfraDownloadImageResourceRequest
infraDownloadImageResourceRequest
                                                        ,
final BindingResult bindingResult) throws TelewebApiException
    {
        ...
        //응답
        return ResponseEntity.ok()
                            .contentType(mediaType)

.body(fileDownloadResponse.getResource());
    }
}
```

        - 없는 파일인 경우 HttpStatus.NO_CONTENT 리턴함 (클라이언트에서는 아무런 반응 없음)

- 참고 : 게시물 팝업-상세 / 게시물 팝업-처리
    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/board/web/BoardPopupController.java

```
public class BoardPopupController
{
    ...
    @ApiOperation(value = "게시물 팝업-처리",
                  notes = "처리팝업으로 이동한다")
    @GetMapping("/board/web/board-process-popup")
    public String moveBoardProcessPopup(@RequestParam("BRD_ID") String
brdId, Model model) throws TelewebWebException
    {
        log.debug("moveBoardProcessPopup");

        //이미지관리일 경우
        if("4".equals(brdId)) {
            //파일 속성
            final FilePropertiesResponse fileProperties =
fileRulePropertiesUtils.getProperties(RepositoryBusiType.chat,
RepositoryPathType.images); //채팅 > 이미지
            log.debug("fileProperties>>>{}", fileProperties);
            model.addAttribute("fileProperties", fileProperties);
        }
        else {
            //파일 속성
            final FilePropertiesResponse fileProperties =
fileRulePropertiesUtils.getProperties(RepositoryBusiType.bbs,
RepositoryPathType.files); //게시판 > 파일
            log.debug("fileProperties>>>{}", fileProperties);
            model.addAttribute("fileProperties", fileProperties);
        }

        //에디터 속성
        final EditorPropertiesResponse editorProperties =
editorRulePropertiesUtils.getProperties(RepositoryBusiType.bbs,
RepositoryPathType.images); //게시판 > 이미지
        log.debug("editorProperties>>>{}", editorProperties);

        model.addAttribute("editorProperties", editorProperties);

        model.addAttribute("timestamp", DateCmmnUtils.toEpochMilli());
        return "board/board-process-popup";
    }
}
```

- ~/palette-hkcloud-rnd-webm-common/src/main/resources/templates/board/board-process-popup.html

```
<!doctype html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
...
<!--/* 첨부파일 header 관련 include */-->
<th:block th:if="${fileProperties.enabled == true}"
th:include="fragments/file/file-headerinc"></th:block>

<!--/* 에디터 header 관련 include */-->
<th:block th:if="${editorProperties.enabled == true}"
th:include="fragments/editor/editor-headerinc"></th:block>

<!-- 해당 화면 script파일 include -->
<script src="/board/board-process-popup.js"></script>
...
<body class="frame-popup">
    ...
            <!--/* 첨부파일 그리드 */-->
            <th:block th:if="${fileProperties.enabled == true}"
th:include="fragments/file/file-bodyinc-grid"></th:block>
            <div class="tt-btn-area">
                <button type="button" class="tt-btn is-md is-main"
id="btnSave">저장</button>
            </div>
    ...
    <!--/* 첨부파일 다이얼로그 */-->
    <th:block th:if="${fileProperties.enabled == true}"
th:include="fragments/file/file-bodyinc-dialog"></th:block>
</body>
</html>
```

- ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/board/board-process-popup.js

```
...
/**********************************************************************
**********
 * Document Ready : jquery에서 제공하는 함수를 이용하여 화면이 로드될 때 처
리할 함수를 정의한다.
**********************************************************************
*********/
function domReady(){
    //첨부파일 초기화 처리
    FileAtchmnfl.init();
    ...
};
...
var BoardProcessPopup={
```

```
    ...
    selectRtn:function(){
        ...
        //파라메터정의
        var objJsonParams = new TelewebJson();
        ...

        //서비스 호출
        var objJsonReturn = $.bizServiceCall(objJsonParams);

        //결과값 반환
        if(!objJsonReturn.getErrorFlag()){
            ...
            //첨부파일조회

FileAtchmnflDownload.selectFiles(objJsonReturn.getData("FILE_GROUP_KEY"
));
            ...
        }
        ...
    },
    ...
}
```

5.11. Editor 관련

- ckeditor 4.7 js lib 사용됨
- 구조
  - 설정
    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml

      ```
      on-profile: "editor"
      ```

    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/properties/editor/EditorProperties.java

      ```
      ```

  - JAVA package
    - ~~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/editor/**
  - HTML
    - 에디터 자바스크립트 설정 및 에디터 로딩
      - ~/palette-hkcloud-rnd-webm-common/src/main/resources/templates/fragments/editor/editor-headerinc.html

        ```
        ```

- JS
  - 에디터 코어
    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/editor/editor-1.0.0.js

      ```
      ```

  - CKEDITOR
    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/editor/editor-ckeditor-1.0.0.js

      ```
      ```

- 게시판 처리 참고
  - ~/palette-hkcloud-rnd-webm-common/src/main/resources/templates/board/board-process-popup.html

    ```
    <!--/* 에디터 header 관련 include */-->
    <th:block th:if="${editorProperties.enabled == true}"
    th:include="fragments/editor/editor-headerinc"></th:block>
    ```

  - ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/board/board-process-popup.js

    ```
    var editorConfig = {
        editorObjId: "BRD_RMK",
        callbackFn: null
    };
    Editor.init(editorConfig);
    ```

5.12. 게시판 관련

- 게시판 관리에 게시판 형태 추가됨
  - 실시간/이미지/일반

5.14. JS console 관련

- 개발/운영 모드에 따른 log 처리
  - 설정
    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml

```
  ######################################################
  #
  # 팔레트(palette) 속성
  # ----------------------------------------------------
  # include 'palette' profile
  ######################################################
  ---
  spring.config.activate:
    on-profile: "palette"
  # ----------------------------------------
  # PaletteProperties
  # ----------------------------------------
  palette:
    service-mode: DEV
    ...
```

    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/static/common/js/TwbCommon.js

```
  // 콘솔 로그 처리
  var consoleHolder = console;
  function debug(bool){if(!bool){consoleHolder=console;console=
{};Object.keys(consoleHolder).forEach(function(key)
{console[key]=function(){};})}else{console=consoleHolder;}}
  ...
  //개발/운영 체크하여 동작여부 결정함
  if(G_SVR_SVC_MODE === "DEV") { debug(true); }else{
debug(false);}
```

6. 가이드

6.1. 개발 패키지 설치 가이드

- ~/doc/PALETTE-DEV-INSTALL-GUIDE.pdf 파일 참조

6.2. 데이터소스 추가 가이드

- **SMS 발송 시스템 데이터 소스 추가 예시임.**
- 데이터 소스 속성에 시스템명 추가
    - ~/palette-hkcloud-rnd-webm-common/src/main/resources/application-local.yml

```
##########################################################
#
# 데이터소스(datasources) 속성
# ㄴ명명 규칙: datasources.[시스템명].[master or slave].[각 속성]
# --------------------------------------------------
# include 'datasources' profile
##########################################################
---
spring.config.activate:
on-profile: "datasources"
# ----------------------------------------
# DatasourcesPaletteProperties
# ----------------------------------------
datasources:
  datasource:
    default-pool-name: "palette"
    #팔레트
    palette:
      db-vendor: "Oracle"
      master:
        jndi-name: "PALETTE_MASTER_DS"
        expected-type: javax.sql.Datasource
      slave:
        jndi-name: "PALETTE_SLAVE_DS"
        expected-type: javax.sql.Datasource
    #SMS 발송 시스템
    sms:
      #해당 vendor명은 Mybatis dababaseId와 매핑되어야 함
      db-vendor: "Mysql"
      master:
        jndi-name: "SMS_MASTER_DS"
        expected-type: javax.sql.Datasource
        jdbc-url:
ENC(dq2iz3tcpVULlmLIZHGENQ0hFFj0a7WV2SLqeqVgNrUrXX3GbOrsE3fXGYzPpYhGjfj
SB4HQmsy7zapyHI8d4C8gKSedJgUls5m5wEWhwRY=)
        username:
ENC(5inANxK1AMUl4G8JU/1tM+zf8yDlprK90+CI+ZXAvDf89EKZZniYoo1nceWwtnc0)
        password:
```

```
ENC(vgus9OimxxxBJW+YU7e69Y1LrufVRvEJNzuyA5hGKU1wTC5cTMiTKMZZ1c33dXdE)
          ...
        slave:
          jndi-name: "SMS_SLAVE_DS"
          expected-type: javax.sql.Datasource
          jdbc-url:
ENC(dq2iz3tcpVULlmLIZHGENQ0hFFj0a7WV2SLqeqVgNrUrXX3GbOrsE3fXGYzPpYhGjfj
SB4HQmsy7zapyHI8d4C8gKSedJgUls5m5wEWhwRY=)
          username:
ENC(5inANxK1AMUl4G8JU/1tM+zf8yDlprK90+CI+ZXAvDf89EKZZniYoo1nceWwtnc0)
          password:
ENC(vgus9OimxxxBJW+YU7e69Y1LrufVRvEJNzuyA5hGKU1wTC5cTMiTKMZZ1c33dXdE)
          ...
```

- 프로퍼티 자바 Config 추가
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/properties/datasources/DatasourcesProperties.java

```
/**
* 데이터소스 팔레트 속성
* @author RND
*
*/
@Getter
@RequiredArgsConstructor
@Validated
@ConstructorBinding
@ConfigurationProperties(prefix = "datasources")
public class DatasourcesProperties
{
    private final Datasource datasource;

    //데이터소스 속성
    @Getter
    @RequiredArgsConstructor
    public static final class Datasource
    {
        @NotNull
        private final DatasourcePoolName defaultPoolName;

        //팔레트 데이터소스
        private final Palette palette;

        @Getter
        @RequiredArgsConstructor
        public static final class Palette
        {
            @NotNull
            private final DatasourceDbVendor dbVendor;
            ...
```

```
            }

            //SMS 발송 시스템
            private final Sms sms;

            @Getter
            @RequiredArgsConstructor
            public static final class Sms
            {
                @NotNull
                private final DatasourceDbVendor dbVendor;
//DatasourceDbVendor에 정의되지 않는 dbVendor인 경우 추가해줘야 함
                ...
            }
        }
    }
```

- 데이터소스 POOL NAME ENUMER 추가
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/datasources/enumer/DatasourcePoolName.java

```
/**
* 데이터소스 POOL NAME
* ㄴ 다중으로 만드는 경우 함께 추가
* @author leeiy
*
*/
public enum DatasourcePoolName {
    //팔레트
    palette,
    //SMS 발송 시스템
    sms
}
```

- SMS 발송 시스템용 데이터소스 패키지 생성
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/datasources/datasource/sms
    - SmsConnMapper.java
    - SmsDataSourceNameList.java
    - SmsReplicationRoutingDataSource.java
    - SmsRoutingDataSourceConfig.java
- 추가된 속성에 맞게 수정
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/config/datasources/datasource/sms/SmsRoutingDataSourceConfig.java

```
/**
  * SMS 발송 시스템 라우팅 데이터소스 설정 - JNDI 지원
  * @author 이일용
  *
  */
  @Slf4j
  @Configuration
  @RequiredArgsConstructor
  @MapperScan(value = "kr.co.hkcloud.palette",
              annotationClass = SmsConnMapper.class,
              sqlSessionFactoryRef = "smsRoutingSqlSessionFactory")
  @EnableTransactionManagement
  public class SmsRoutingDataSourceConfig
  {
      private final DatasourcesProperties     datasourceProperties;
      private final VendorDatabaseIdProperies
vendorDatabaseIdProperies;
      private final PaletteProfileUtils       paletteProfileUtils;
      private JndiDataSourceLookup lookup = new JndiDataSourceLookup();

      /**
      * Master 데이터소스 -JNDI를 사용하고자 하면 해당 개발단계 profile을
수정하여 적용할 것
      *
      * @return DataSource
      */
      @Primary
      @Bean(destroyMethod = "")  //dbvendor에 따라 조금씩 다를 수 있음
      @ConfigurationProperties(prefix =
"datasources.datasource.sms.master")
      public DataSource smsRoutingMasterDataSource()
      {
          PaletteProfiles profile =
paletteProfileUtils.getActiveProfile();
          log.trace("*sms master datasource profile={}", profile);
          switch(profile)
          {
              ...
          }
      }


      /**
      * Slave 데이터소스 -JNDI를 사용하고자 하면 해당 개발단계 profile을 수
정하여 적용할 것
      *
      * @return DataSource
      */
      @Bean(destroyMethod = "")  //dbvendor에 따라 조금씩 다를 수 있음
      @ConfigurationProperties(prefix =
"datasources.datasource.sms.slave")
      public DataSource smsRoutingSlaveDataSource()
      {
```

```
            PaletteProfiles profile =
paletteProfileUtils.getActiveProfile();
            log.trace("*sms slave datasource profile={}", profile);
            switch(profile)
            {
                ...
            }
        }


        /**
        * @param  routingMasterDataSource
        * @param  routingSlaveDataSource
        * @return
        */
        @Bean
        public DataSource smsRoutingDataSource()
        {
            Map<Object, Object> targetDataSourceMap = new
LinkedHashMap<Object, Object>();
            targetDataSourceMap.put("master",
smsRoutingMasterDataSource());
            targetDataSourceMap.put("slave_1",
smsRoutingSlaveDataSource());
            SmsReplicationRoutingDataSource routingDataSource = new
SmsReplicationRoutingDataSource();
            routingDataSource.setTargetDataSources(targetDataSourceMap);

routingDataSource.setDefaultTargetDataSource(smsRoutingMasterDataSource
());
            return routingDataSource;
        }


        /**
        * @param  routingDataSource
        * @return
        */
        @Bean
        public DataSource smsRoutingLazyDataSource()
        {
            return new
LazyConnectionDataSourceProxy(smsRoutingDataSource());
        }


        /**
        * @param  routingLazyDataSource
        * @param  applicationContext
        * @return
        * @throws Throwable
        */
        @Bean
        public SqlSessionFactory
```

```
smsRoutingSqlSessionFactory(ApplicationContext applicationContext)
throws Throwable
      {
          String locationPattern = "";

          DatabaseIdProvider vendorDatabaseIdProvider = new
VendorDatabaseIdProvider();

vendorDatabaseIdProvider.setProperties(vendorDatabaseIdProperies.getDbV
endorProperties());

          DatasourceDbVendor dbVendor =
DatasourceDbVendor.valueOf(vendorDatabaseIdProvider.getDatabaseId(smsRo
utingLazyDataSource()));
          log.debug("dbVendor=============> {}", dbVendor);
          switch(dbVendor)
          {
              case Oracle:
              {
                  locationPattern =
"classpath*:kr/co/hkcloud/palette/**/dao/xml/*Mapper_Oracle.xml";
                  break;
              }
              case Mysql:
              {
                  locationPattern =
"classpath*:kr/co/hkcloud/palette/**/dao/xml/*Mapper_Mysql.xml";
                  break;
              }
              default:
              {
                  locationPattern =
"classpath*:kr/co/hkcloud/palette/**/dao/xml/*Mapper_Mysql.xml";
                  break;
              }
          }

          final SqlSessionFactoryBean sessionFactory = new
SqlSessionFactoryBean();
          sessionFactory.setDataSource(smsRoutingLazyDataSource());

sessionFactory.setTypeHandlersPackage("kr.co.hkcloud.palette");  //TODO
데이터소스 TypeHandlers 테스트중 20210610

sessionFactory.setTypeAliasesPackage("kr.co.hkcloud.palette");

sessionFactory.setDatabaseIdProvider(vendorDatabaseIdProvider);

sessionFactory.setMapperLocations(applicationContext.getResources(locat
ionPattern));
          return sessionFactory.getObject();
      }
```

```
        /**
        * @param  paletteRoutingSqlSessionFactory
        * @return
        */
        @Primary
        @Bean
        public SqlSessionTemplate
smsRoutingSqlSessionTemplate(@Qualifier("smsRoutingSqlSessionFactory")
SqlSessionFactory smsRoutingSqlSessionFactory)
        {
            org.apache.ibatis.session.Configuration mybatisConfig =
smsRoutingSqlSessionFactory.getConfiguration();
            ...

            return new SqlSessionTemplate(smsRoutingSqlSessionFactory);
        }


        @Bean
        public PlatformTransactionManager transactionManager()
        {
            return new
DataSourceTransactionManager(smsRoutingLazyDataSource());
        }
    }
```

- TwbComDao를 사용하는 경우
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/common/twb/dao/TwbComDAO.java

```
    @Slf4j
    @Repository
    @Transactional
    public class TwbComDAO
    {
        private SqlSession paletteRoutingSqlSessionTemplate;  //팔레트
        private SqlSession smsRoutingSqlSessionTemplate;  //SMS 발송 시스
템

        @Autowired
        @Qualifier("paletteRoutingSqlSessionTemplate")
        public void setPaletteRoutingSqlSessionTemplate(SqlSession
paletteRoutingSqlSessionTemplate)
        {
            this.paletteRoutingSqlSessionTemplate =
paletteRoutingSqlSessionTemplate;
        }

        @Autowired
        @Qualifier("smsRoutingSqlSessionTemplate") //SMS 발송 시스템 Bean
        public void setSmsRoutingSqlSessionTemplate(SqlSession
```

```
smsRoutingSqlSessionTemplate)
	{
		this.smsRoutingSqlSessionTemplate =
smsRoutingSqlSessionTemplate;
	}
	...

	public TelewebJSON select(String sqlNameSpace, String sqlNm,
TelewebJSON acJson) throws TelewebDaoException
	{
		JSONObject obj =
JSONObject.fromObject(acJson.getHeaderObject());

		if(obj.containsKey("POOL_NM")) {
			DatasourcePoolName datasourcePoolName =
DatasourcePoolName.valueOf(obj.getString("POOL_NM"));
			return this.select(sqlNameSpace, sqlNm, acJson, null,
datasourcePoolName);
		}
		else {
			return this.select(sqlNameSpace, sqlNm, acJson, null,
DatasourcePoolName.palette);
		}
	}
	/**
	* sqlSession을 변환한다.
	*
	* @param  map                Map<String, Object>.
	* @return                    String.
	* @throws TelewebDaoException
	*/
	private SqlSession getSqlSession(HttpServletRequest objRequest,
DatasourcePoolName datasourcePoolName) throws TelewebDaoException
	{
		SqlSession sqlSession = null;
		switch(datasourcePoolName)
		{
			//팔레트
			case palette:
			{
				sqlSession = this.paletteRoutingSqlSessionTemplate;
				break;
			}
			//SMS 발송 시스템
			case sms:
			{
				sqlSession = this.smsRoutingSqlSessionTemplate;
				break;
			}
		}
		return sqlSession;
	}
  }
```

- 해당 업무의 ~/dao/xml/Sms**Mapper_Mysql.xml 생성

```
  <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

  <mapper
namespace="kr.co.hkcloud.palette.infra.sms.dao.InfraSmsMapper">
  <!-- 조회 -->
  <select id="selectRtn"  parameterType= "java.util.HashMap"
resultType="java.util.HashMap">
  ...
  </select>
  </mapper>
```

- 사용법
  - TwbComDAO 클래스에서 TelewebJSON에 **POOL_NM** 키가 있는 경우 데이터소스를 지정해 줄 수 있다.

```
jsonParams.setString("POOL_NM", "sms");  //데이터소스를 SMS 발송 시
스템으로 지정
objRetParams =
mobjDao.select("kr.co.hkcloud.palette.infra.sms.dao.InfraSmsMapper
", "selectRtn", jsonParams);
```

- 어노테이션을 사용하는 경우
  - Mybatis Mapper 인터페이스 생성
    - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/infra/sms/dao/FileDbMngMapper.java

```
@SmsConnMapper  //SMS 발송 시스템 데이터소스로 지정
public interface SmsDbMngMapper
{
    @SelectProvider(value = SmsDbMngSelectMysqlSqlProvider.class,
                    method = "selectRtn",
                    databaseId = "Mysql")
    SmsDbMngSelectResponse
selectRtn(@Validated(SmsDbMngRequest.GroupServiceSelectTargetType.
class) final SmsDbMngSelectRequest smsDbMngSelectRequest) throws
PaletteDaoException;
}
```

- 테이블 관련 Enum 생성
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/infra/sms/dao/enumer/SMS_DB_MNG.java

```
/**
* SMS DB 관리 관련 테이블과 컬럼 정의
* ㄴ Rule : 케밥케이스, 모두 대문자로 정의
* @author leeiy
*/
public final class SMS_DB_MNG
{
    //테이블
    public enum TABLE
    {
        //SMS 발송 시스템
        PLT_SMS
    }

    //컬럼정의
    public static final class COLUMN
    {
        //SMS 발송 시스템
        public enum PLT_SMS
        {
            ASP_CUST_KEY        //ASP_고객사_키
            , FILE_GROUP_KEY      //파일그룹키
            ...
        }
    }
}
```

- SqlProvider 생성
  - ~/palette-hkcloud-rnd-webm-common/src/main/java/kr/co/hkcloud/palette/infra/sms/dao/provider/SmsDbMngSelectMysqlSqlProvider.java

```
//@formatter:off
/**
* SMS DB 관리 조회 Mysql SQL Provider
* Implements the ProviderMethodResolver on your provider class
*
* @author leeiy
*
*/
public class SmsDbMngSelectMysqlSqlProvider implements
ProviderMethodResolver
{
    ...

    public static String selectRtn(final SmsDbMngSelectRequest
```

```
smsDbMngSelectRequest)
    {
        return new SQL() {{
            SELECT(

SMS_DB_MNG.COLUMN.PLT_SMS.FILE_GROUP_KEY.name()  //-- 파일그룹키
                  , SMS_DB_MNG.COLUMN.PLT_SMS.FILE_KEY.name()
//-- 파일키
                  ...
            );
            FROM(
                    SMS_DB_MNG.TABLE.PLT_SMS.name()
            );
            WHERE(
                    String.format("%s = #{aspCustKey}"  ,
SMS_DB_MNG.COLUMN.PLT_SMS.ASP_CUST_KEY.name())
                  , String.format("%s = #{fileGroupKey}",
SMS_DB_MNG.COLUMN.PLT_SMS.FILE_GROUP_KEY.name())
                  ...
            );

            //업무유형

if(!StringUtils.isEmpty(smsDbMngSelectRequest.getBusiType()))
            {
                WHERE(String.format("%s = #{busiType}",
SMS_DB_MNG.COLUMN.PLT_SMS.BUSI_TYPE.name()));
            }
            ...
        }}.toString();
    }
    ...
}
```

- 사용법
  - 클래스에 주입

```
private final SmsDbMngMapper smsDbMngMapper;
```

  - 함수에서 사용

```
final SmsDbMngSelectResponse response =
smsDbMngMapper.selectRtn(smsDbMngSelectRequest);
```

6.3. 빌드(Gradle) 가이드

- 팔레트는 모듈별로 분리됨에 따라 기존 텔레톡 build 방식과 다름
- 구조
  - 부모 프로젝트
    - palette-hkcloud-rnd-webm
    - 주요 설정
      - ~/palette-hkcloud-rnd-webm/build.gradle

```
buildscript {
    ...
}
allprojects {
    group = 'kr.co.hkcloud.palette'
    //빌드 시 prefix로 붙는다
    version = '2.0.100-SNAPSHOT'
}
subprojects {
    ...
    configurations {
        ...
        // exclude Tomcat: 내장 was undertow로 기동하기 위해 설
정
        compile.exclude module: 'spring-boot-starter-tomcat'
    }
    ...
    //java 영역안에 xml 파일을 두기 위해 설정
    sourceSets.main.resources {
        srcDirs += ["src/main/java"];
        exclude "**/*.java"
    }
    ...
    dependencies {
        //라이브러리들...
        ...
    }
    ...
}
```

      - ~/palette-hkcloud-rnd-webm/settings.gradle

```
rootProject.name = 'palette-hkcloud-rnd-webm'
include 'palette-hkcloud-rnd-webm-common'
include 'palette-hkcloud-rnd-webm-login'
include 'palette-hkcloud-rnd-webm-phone'
include 'palette-hkcloud-rnd-webm-chat'
include 'palette-hkcloud-rnd-webm-km'
include 'palette-hkcloud-rnd-webm-run'
```

- 공통 모듈 서브 프로젝트
  - palette-hkcloud-rnd-webm-common
  - 주요 설정
    - ~/palette-hkcloud-rnd-webm-common/build.gradle

```
//jar 활성
jar {
    enabled = true
}
//bootjar 비활성
bootJar {
    enabled = false
}
...
```

- 로그인 모듈 서브 프로젝트
  - palette-hkcloud-rnd-webm-login
  - 주요 설정
    - ~/palette-hkcloud-rnd-webm-login/build.gradle

```
//jar 활성
jar {
    enabled = true
}
//bootjar 비활성
bootJar {
    enabled = false
}
dependencies {
    //공통과 채팅 모듈 프로젝트를 참조.
    implementation project(":palette-hkcloud-rnd-webm-
common")
    //채팅 모듈은 로그인 / 로그아웃 시 상담원 상태를 정리해 주기
위함이다.
    implementation project(":palette-hkcloud-rnd-webm-chat")
    //include  palette-web-km custom lib
    compile fileTree(dir: 'libs-palette-webm-login', include:
['*.jar'])
}
```

- 전화 모듈 서브 프로젝트
  - palette-hkcloud-rnd-webm-phone
  - 주요설정
    - ~/palette-hkcloud-rnd-webm-phone/build.gradle

```
//jar 활성
jar {
    enabled = true
}
//bootjar 비활성
bootJar {
    enabled = false
}
dependencies {
    //공통 모듈 프로젝트를 참조 .
    implementation project(":palette-hkcloud-rnd-webm-
common")
    //include palette-web-phone custom lib
    compile fileTree(dir: 'libs-palette-webm-phone', include:
['*.jar'])
}
```

- 채팅 모듈 서브 프로젝트
  - palette-hkcloud-rnd-webm-chat
  - 주요 설정
    - ~/palette-hkcloud-rnd-webm-chat/build.gradle

```
//jar 활성
jar {
    enabled = true
}
//bootjar 비활성
bootJar {
    enabled = false
}
dependencies {
    //공통 모듈 프로젝트를 참조.
    implementation project(":palette-hkcloud-rnd-webm-
common")
    //include palette-web-chat custom lib
    compile fileTree(dir: 'libs-palette-webm-chat', include:
['*.jar'])
}
```

- 지식 모듈 서브 프로젝트
  - palette-hkcloud-rnd-webm-km
  - 주요 설정
    - ~/palette-hkcloud-rnd-webm-km/build.gradle

```
//jar 활성
jar {
    enabled = true
}
```

```
//bootjar 비활성
bootJar {
    enabled = false
}
dependencies {
    //공통 모듈 프로젝트를 참조.
    implementation project(":palette-hkcloud-rnd-webm-
common")
    //include palette-web-km custom lib
    compile fileTree(dir: 'libs-palette-webm-km', include:
['*.jar'])
}
```

- 기동 모듈 서브 프로젝트(Spring boot dashbord, gradle build)
  - palette-hkcloud-rnd-webm-run
  - 주요 설정
    - ~/palette-hkcloud-rnd-webm-run/build.gradle

```
//jar 비활성
jar {
    enabled = false
}
//bootjar 활성
bootJar {
    enabled = true
}
dependencies {
    //공통 모듈 프로젝트를 참조
    implementation project(":palette-hkcloud-rnd-webm-
common")
    //로그인 모듈 프로젝트를 참조
    implementation project(":palette-hkcloud-rnd-webm-login")
    //전화 모듈 프로젝트를 참조
    implementation project(":palette-hkcloud-rnd-webm-phone")
    //채팅 모듈 프로젝트를 참조
    implementation project(":palette-hkcloud-rnd-webm-chat")
    //지식 모듈 프로젝트를 참조
    implementation project(":palette-hkcloud-rnd-webm-km")
}
```

  - spring boot dashboard 기동 시 사용한다.
    - 오른쪽버튼 > Open Config > Arguments 탭 > VM arguments 영역

```
-Djava.net.preferIPv4Stack=true -
Dspring.application.name=palette-webm1 -Dserver.port=8443 -
Dspring.profiles.active=local,local-phone,local-chat,local-km
-Djasypt.key=paletteHello -Dpalette.chat.cipher=Y -
Dpalette.chat.key=MnR6bzI4bjdocGEw
```

        - 각 Arguments는 '**Spring boot 부트 시 JVM Arguments**' 장 참조
- 로컬 빌드
    - STS로 빌드
        - 빌드 작업
            - STS > Window > Show View > Other > Gradle > Gradle Tasks
                - palette-hkcloud-rnd-webm > build > Clean 클릭
                - palette-hkcloud-rnd-webm > build > build 클릭
        - 빌드 진행 상황
            - STS > Window > Show View > Other > Gradle > Gradle Executions
                - Run build
    - CLI로 빌드 (Cmd 참 사용)
        - 빌드 및 진행상황(console log)
            - ~/palette-hkcloud-rnd-webm-run/gradlew.bat clean build
- 로컬 빌드 완료 시
    - ~/palette-hkcloud-rnd-webm-run/build/libs
        - palette-hkcloud-rnd-webm-run-2.0.100-SNAPSHOT.jar
- 오프라인 빌드 (기본 세팅은 되어 있음)
    - gradle repository에 프로젝트에서 사용하는 모든 종속라이브러리들이 존재해야 한다.
        - D:\PALETTE\hkcloud\rnd\repository\gradle
        - D:\PALETTE\hkcloud\rnd\repository\gradle\caches\modules-2\files-2.1
    - gradle.user.home 경로 체크
        - ~/palette-hkcloud-rnd-webm/gradle.properties

```
systemProp.gradle.user.home=D:/PALETTE/hkcloud/rnd/repository/grad
le
```

    - wrapper bin 파일 체크
        - ~/palette-hkcloud-rnd-webm/gradle/wrapper/gradle-wrapper.properties

```
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
#distributionUrl=https\://services.gradle.org/distributions/gradle
-6.9-bin.zip
distributionUrl=file:///D:/PALETTE/hkcloud/rnd/repository/gradle/w
rapper/dists/gradle-6.9-bin/d9tf0js49ceydcud0u9ui929b/gradle-6.9-
bin.zip
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
```

- DEPLOY & BUILD
    - 도커나 젠킨스 등 고객사 배포 및 빌드 환경에 맞게 수정한다.

6.4. Spring boot 기동 절차 가이드

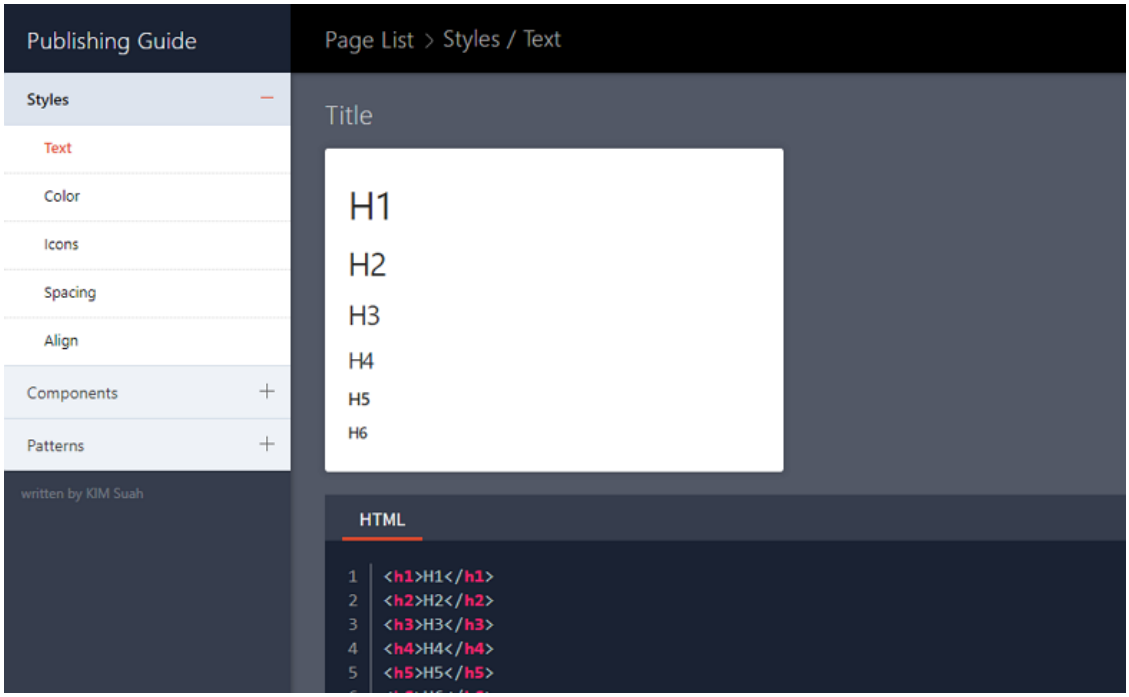
- Spring-boot JVM Args
  - 암호화 키를 생성하려는 경우
    - 무작위 비밀번호 생성 사이트
  - JVM Arguments
    - -Djava.net.preferIPv4Stack=true -Dspring.application.name=palette-webm1 -Dserver.port=8443 -Dspring.profiles.active=local,local-phone,local-chat,local-km -Djasypt.key=paletteHello -Dpalette.chat.cipher=Y -Dpalette.chat.key=MnR6bzI4bjdocGEw
  - JVM Arguments 상세
    - IPv4 지원
      - -Djava.net.preferIPv4Stack=true
    - 어플리케이션 이름
      - -Dspring.application.name=palette-web1
    - 서버 기동 포트
      - -Dserver.port=8443
    - 사용할 모듈 프로필
      - -Dspring.profiles.active=local,local-phone,local-chat,local-km
    - YAML 암호화키
      - -Djasypt.key=paletteHello
    - 채팅-대화내용 암호화 여부
      - -Dpalette.chat.cipher=Y
    - 채팅-대화내용 암호화키-로컬/개발(16)
      - -Dpalette.chat.key=MnR6bzI4bjdocGEw
    - 채팅-대화내용 암호화키-운영(16)
      - -Dpalette.chat.key=MXA2czJuODd3aDQ4
- dashboard
- bootjar

6.5. 퍼블리싱 가이드

- 퍼블리싱 가이드는 2가지로 나뉘어 있다.
  - 1. 기본 프레임워크 퍼블리싱 가이드
  - 2. 채팅상담메인 퍼블리싱 가이드

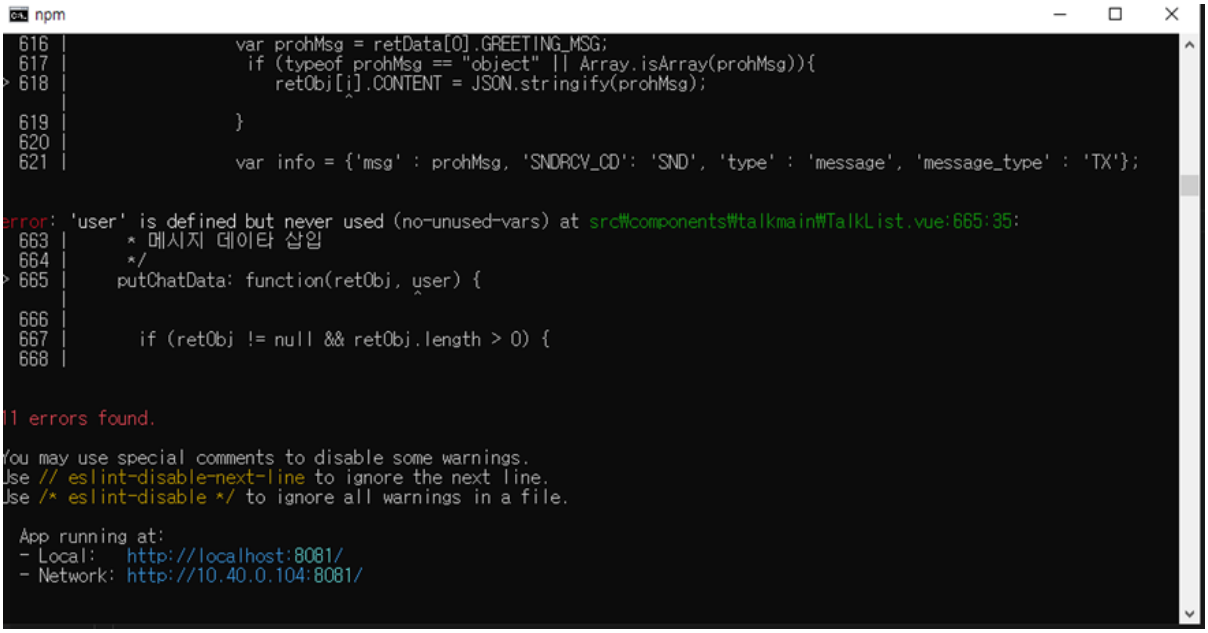
- 기본 프레임워크 퍼블리싱 가이드
  - 퍼블리싱 가이드 페이지는 소스로 구성되어 있다.
  - 가이드 페이지 접속 방법
    - Spring boot 기동 후 팔레트 로그인
    - 초기 접속정보
      - teleweb
      - Test1234!

- http://localhost:8443/common/pbGuide/index.html 에 접속한다.



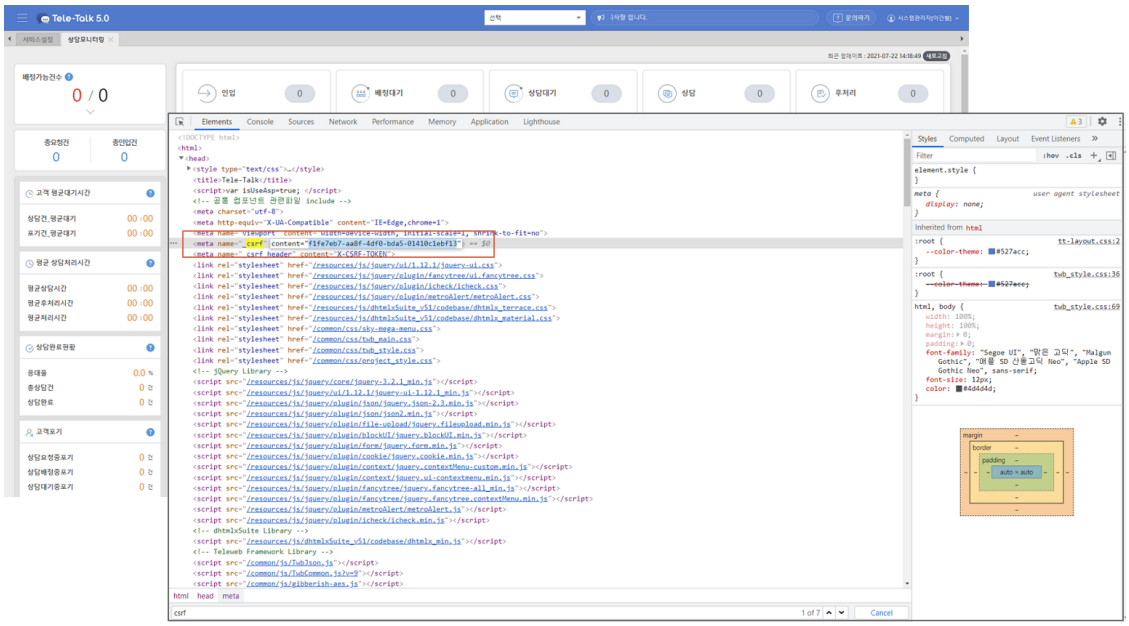
- 채팅상담메인 퍼블리싱 가이드는 문서로 작성되어 있다.
    - 채팅상담메인 frontend 개발환경에서 접속 가능
        - URL : http://localhost:8081/#/pbGuide/index
- 채팅상담메인 개발 가이드
    - 채팅 상담 메인만 vue 프레임워크 개발되어 있다.
    - frontend 패키지는 개발환경을 위한 패키지로, 운영 서버에는 배포되면 안된다.
    - frontend 패키지를 통해 build된 결과물만 서버 소스에 포함한다.
    - 채팅상담메인 개발환경 설치 및 실행
        - ~/doc/PALETTE-DEV-INSTALL-GUIDE.pdf 파일 참조
        - VS Code에 설치된 기본 extension(확장기능)이 있기 때문에, 전달한 VS Code 파일로 설치할 것을 권장함

- run_dev.bat 실행 (개발환경 실행)



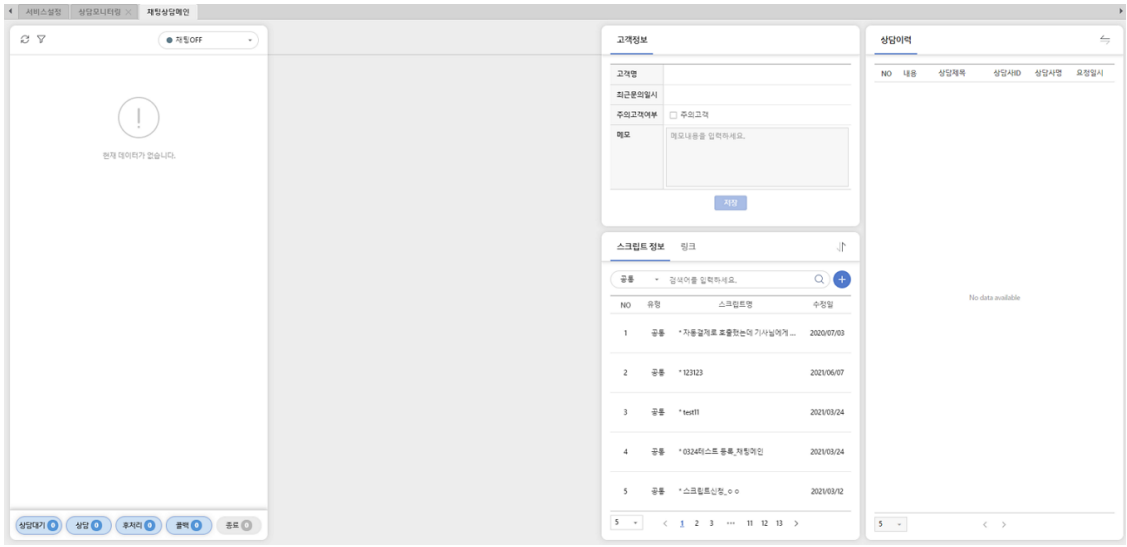
- frontend 개발환경 접속 방법

1. 팔레트 서버가 구동
2. 팔레트 서버 로그인
3. frontend 개발환경 실행 (run_dev.bat)
4. 팔레트 브라우저에서 F12(디버깅 모드) 실행
5. 팔레트 csrf 값 복사



6. frontend 화면 접속

- URL : http://localhost:8081/#/[위에서 복사한 csrf값]



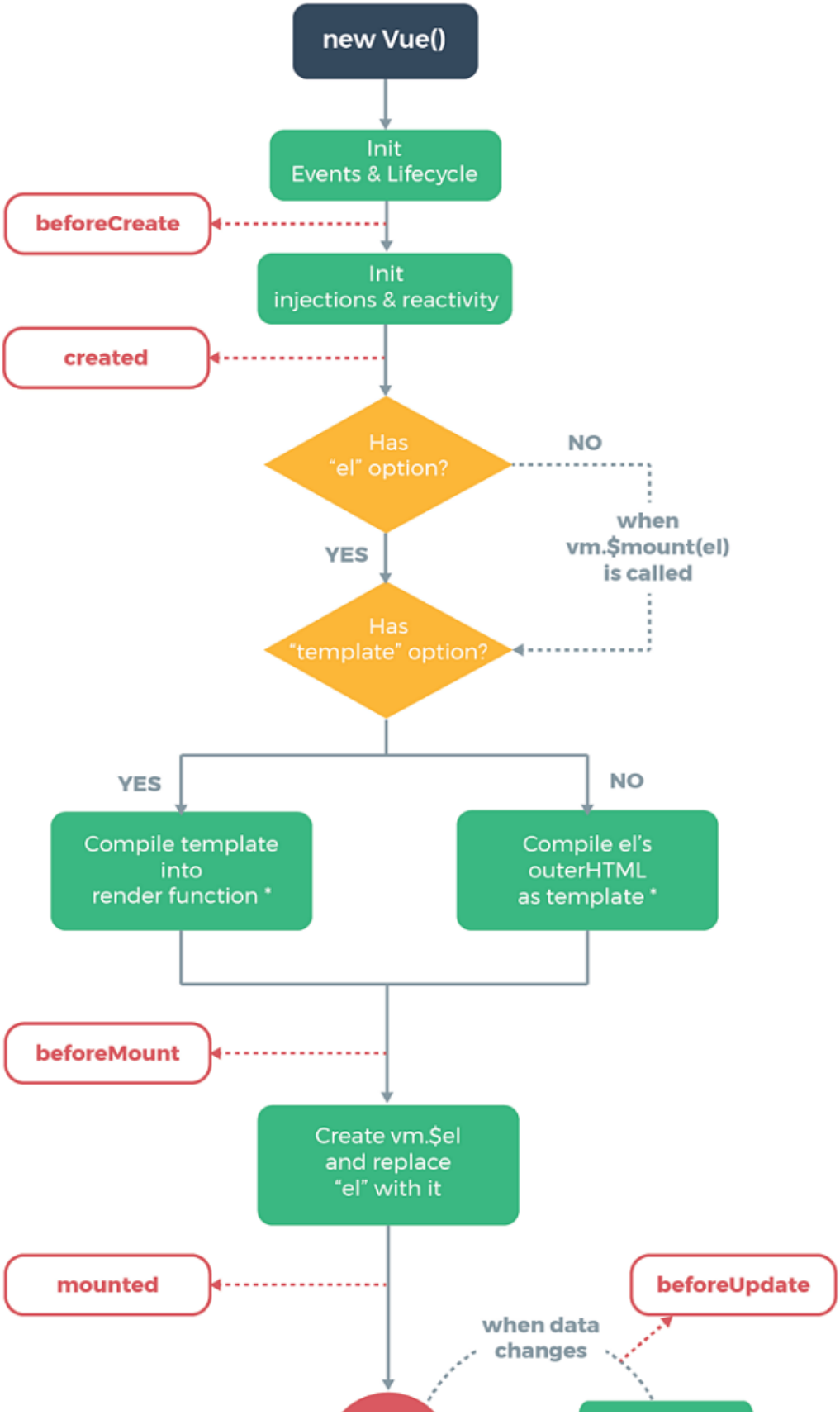
- 패키지 구조

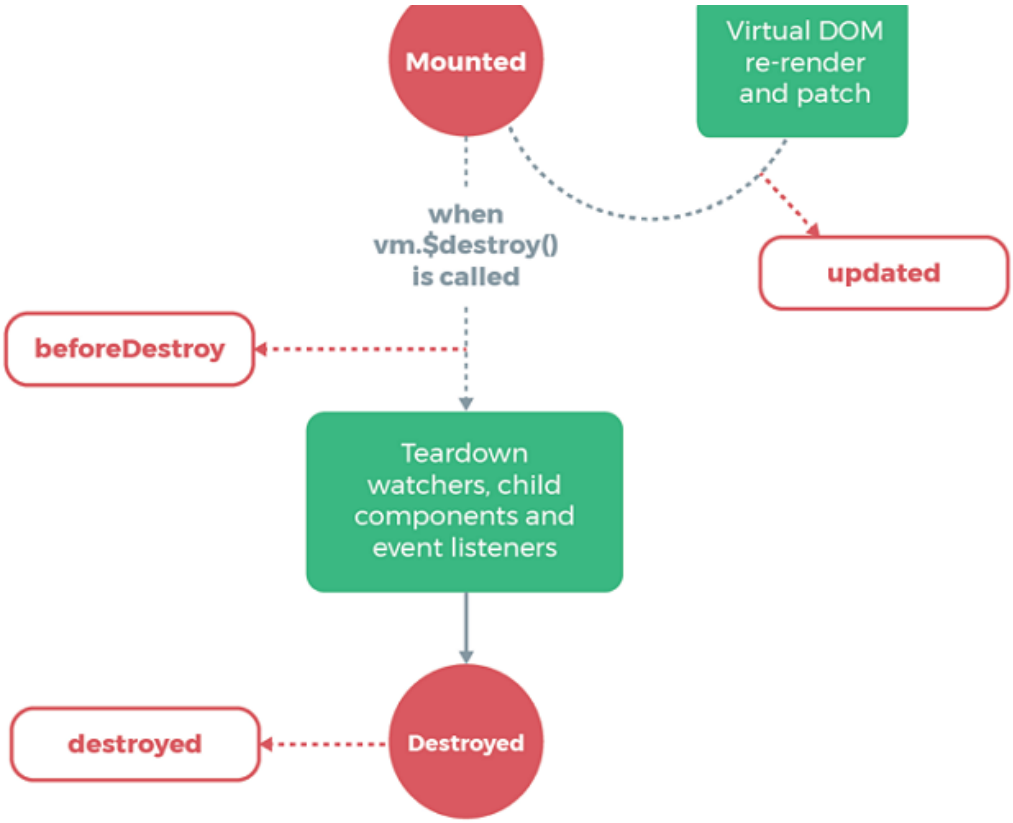
패키지경로	설명
FRONTEND	팔레트 채팅상담메인 FRONTEND 프로젝트
ㄴ .settings	
ㄴ .vscode	
ㄴ build	
ㄴ dist	
ㄴ node_modules	프로젝트에 설치된 nodejs 라이브러리
ㄴ public	
ㄴ src	소스
ㄴ .babelrc	
ㄴ .env.development	development 프로필 설정파일
ㄴ .env.production	production 프로필 설정파일
ㄴ babel.config.js	
ㄴ build.bat	
ㄴ package.json	설치된 라이브러리 목록 및 버전
ㄴ vue.config.js	백엔드 서버정보, 빌드경로, 레파지토리 경로
ㄴ ...	

  - node_modules -vue v2.6.10 (https://kr.vuejs.org/v2/guide/instance.html)
    - 라이프사이클 이해하기 !
    - Vue-cli (프론트 개발용 node 서버 / build 지원)
    - Vuetify 컴포넌트 참조 (https://vuetifyjs.com/ko/getting-started/quick-start/)

- Vuex (https://vuex.vuejs.org/kr/)
- vue-lodash 각종 util (https://lodash.com/docs/)
- Api 통신 Axios (https://www.npmjs.com/package/axios)
- @babel/polyfill (익스에서 es6 문법 적용 가능 하도록)
- Vue router (https://router.vuejs.org/kr/)

- VUE 라이프사이클





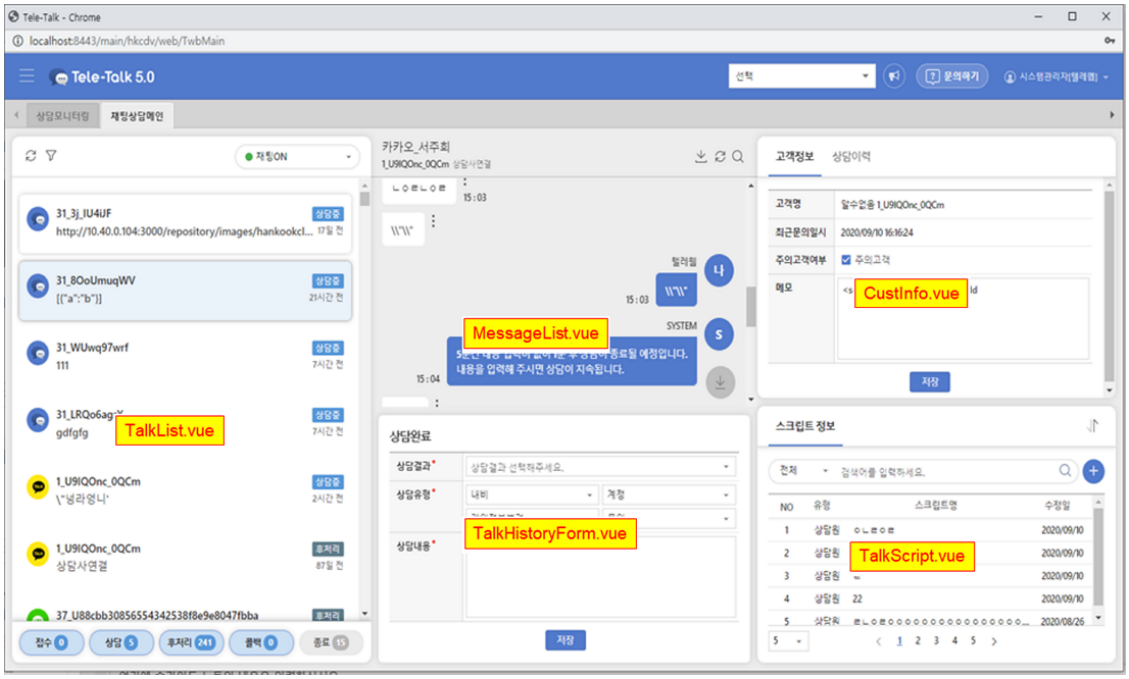
- .env.[프로필] 파일
  - 프로필 설정방법
    - 시스템 환경변수로 프로필 설정함
      - NODE_ENV=development
      - NODE_ENV=production
    - 프로필에 따른 설정파일
      - .env.development
      - .env.production
  - vue 설정값은 반드시 "vue_app_" 로 시작해야 한다.
  - VUE_APP_PRIVATE_KEY : 파라미터 검증 security key
- vue.config.js

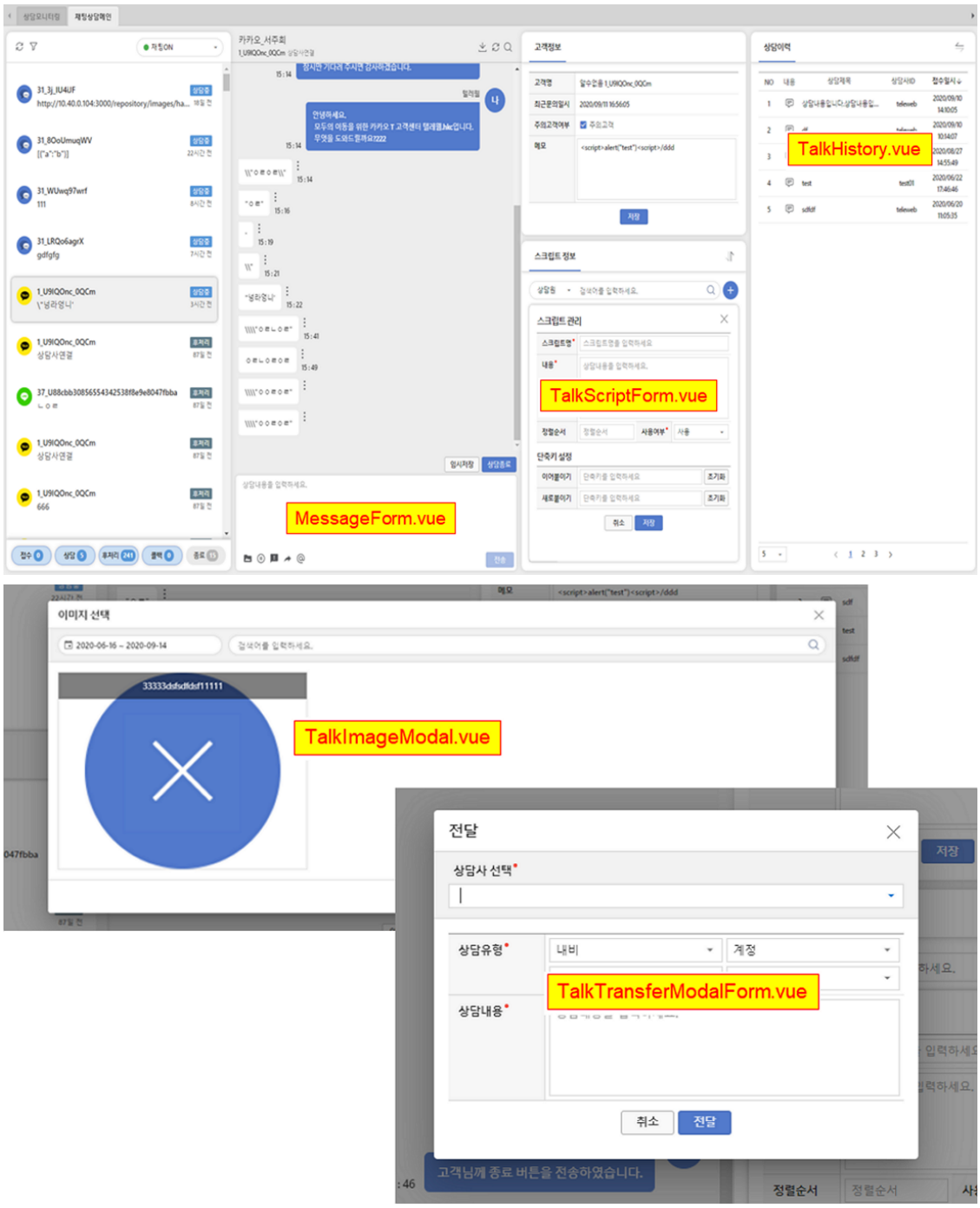
- 개발시에만 사용한다.

- 소스 구조

패키지경로	설명
src	팔레트 채팅상담메인 FRONTEND 프로젝트
└ assets	
└ components	컴포넌트 소스 (.vue) 모든 구성은 컴포넌트로 구성되어 있음
└ — talkmain	
└ — — CustInfo.vue	
└ — — MessageForm.vue	
└ — — ...	
└ mixins	
└ pbGuide	퍼블리싱 가이드 소스
└ plugins	
└ store	메시지 상태 관리 (vuex)
└ utils	
└ — stomp-util.js	웹소켓 연결 소스
└ — tele-api.js	백엔드 API 호출(axios)
└ — event-bus.js	이벤트버스 (컴포넌트간의 이벤트 전달 역할)
└ views	
└ main.js	최초 인입 페이지
└ router.js	페이지 이동

- 화면별 컴포넌트


Tele-Talk - Chrome
localhost:8443/main/hkcdv/web/TwbMain
Tele-Talk 5.0
상담모니터링
채팅상담예언
채팅ON
TalkList.vue
카카오_서주희
MessageList.vue
상담완료
TalkHistoryForm.vue
고객정보
상담이력
CustInfo.vue
저장
스크립트 정보
TalkScript.vue



- 빌드 및 배포
  - 빌드시에 디렉토리를 다 지우고 빌드하기 때문에, 백업이 필요한 경우 빌드 전 백업

- 빌드 경로는 vue.config.js 파일에서 output 에서 설정 가능

- vue 파일이 클 수록 chunk 파일로 쪼개져서 js 로 생성된다.
- 본래 빌드시에는 html 부터 .js까지 모두 생성을 해준다.
- 그러나 채팅상담메인에서는 일부만 vue로 되어 있기 때문에, talkMain.html에 chunk파일을 변경해주어야 한다.
- 파일명 변경이 번거롭지만, 파일명이 바뀌기 때문에, 캐시문제가 어느정도 보완되는 장점도 있다.