UB-Framework ™

|  |
| --- |
| Development of File Attach Guide |

**Version 1.0.0**

**2016-04-01**

**logo_2**

Copyrightⓒ 2016 UBone CO., Ltd. All right reserved.

본 문서는 (주)유비원의 Solution제작 산출물로써 (주) 유비원의 지적 자산이므로

본 문서의 형태 및 내용을 다른 용도로 임의 사용 또는 변경할 수 없음을 밝힙니다.

Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Revision*** | ***Revised Date*** | ***Author*** | ***Authorized***  ***By*** | ***Comments*** | ***Reference*** |
| 1.0 |  | 김병기 |  | 초본 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table of Contents

[Revision History 1](#_Toc447627789)

[Table of Contents 1](#_Toc447627790)

[1. File Attach개요 1](#_Toc447627791)

[1.1. File Attach란? 1](#_Toc447627792)

[1.1.1. 개요 1](#_Toc447627793)

[2. File Attach구성과 흐름 1](#_Toc447627794)

[2.1. File Attach 관련 폴더 구성도 1](#_Toc447627795)

[2.2. SwfUpload 폴더 및 uxl-ext-1.x.x.js 설명 1](#_Toc447627796)

[2.3. File-Attach 테이블 1](#_Toc447627797)

[2.4. File-Attach 기능 과정 및 구조 1](#_Toc447627798)

[2.4.1. File-Attach 실행 과정 1](#_Toc447627799)

[2.4.2. File-Attach 업로드 시 1](#_Toc447627800)

[2.4.3. File-Attach 파일 삭제 시 1](#_Toc447627801)

[2.4.4. File-Attach 파일 다운로드 시 1](#_Toc447627802)

[2.4.5. File-Attach 파일 정보 조회 시 1](#_Toc447627803)

[2.4.6. ServiceSupport 첨부 파일 지원 기능 1](#_Toc447627804)

[3. File Attach 기본적인 사용 1](#_Toc447627805)

[3.1. File Attach 등록 화면 예제 1](#_Toc447627806)

[3.1.1. File Attach 등록 화면 Meta-Studio에서 ATTACH-FILE Control 추가 및 등록 1](#_Toc447627807)

[3.1.2. File Attach 등록 화면 File-control-id 등록 1](#_Toc447627808)

[3.1.3. APMD 서비스 설정 1](#_Toc447627809)

[3.1.4. Jsp 파일에 uxl.ext.Upload 추가 1](#_Toc447627810)

[3.2. File Attach 상세화면 예제 1](#_Toc447627811)

[3.2.1. APMD 서비스 설정(상세화면) 1](#_Toc447627812)

[3.2.2. uxl.ext.upload Option 설정 바꾸기 1](#_Toc447627813)

[3.3. File Attach 수정화면 예제 1](#_Toc447627814)

[3.3.1. APMD 서비스 설정(수정 서비스) 1](#_Toc447627815)

[3.3.2. File Attach 수정화면 예제 1](#_Toc447627816)

[3.4. File Attach 삭제화면 예제 1](#_Toc447627817)

[3.4.1. APMD 서비스 설정(삭제 서비스) 1](#_Toc447627818)

[4. File Attach uxl.ext.Upload부가 옵션 1](#_Toc447627819)

[4.1. uxl.ext.Upload옵션 1](#_Toc447627820)

[5. File Atach 관련 xml 설정 1](#_Toc447627821)

[5.1. Config.xml Upload 설정 1](#_Toc447627822)

[5.2. applicationContext-app-file-policy.xml 설정 1](#_Toc447627823)

# File Attach개요

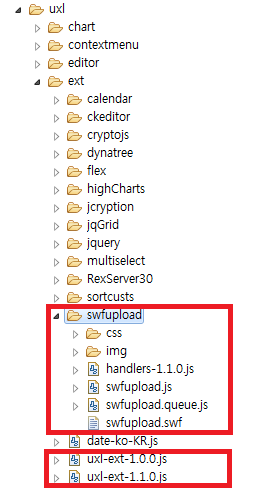
## File Attach란?

### 개요

* 파일 첨부 기능은 기본적으로 UB-Framework에서 지원하는 기능이다.
* Swfupload 라이브러리를 활용하여 비동기식 파일 업로드를 지원하며, 파일 정책에 따른 첨부 파일 관리가 가능하다.

# File Attach구성과 흐름

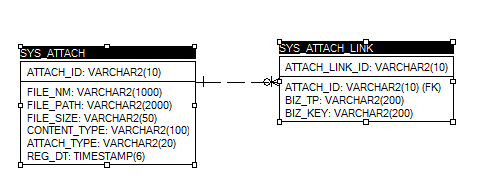
## File Attach 관련 폴더 구성도



## SwfUpload 폴더 및 uxl-ext-1.x.x.js 설명

|  |  |  |
| --- | --- | --- |
| 폴더명 / 파일명 | 용도 | 비고 |
| /Swfupload/handlers-1.1.0.js | File 첨부와 관련된 swfupload.js 에 있는 함수 구현 |  |
| /Swfupload/swupload.js | swfupload.js 플래쉬를 통한 업로드 구현라이브러리 |  |
| /Swfupload/swfupload.queue.js | queue.js |  |
| /Swfupload/swfupload.swf | swfupload.swf |  |
| /Swfupload/css/file.css |  |  |
| /Swfupload/img |  |  |
| /uxl-ext-1.x.x.js | 파일 업로드로 사용한uxl.ext.uploa() 구현 |  |

## File-Attach 테이블



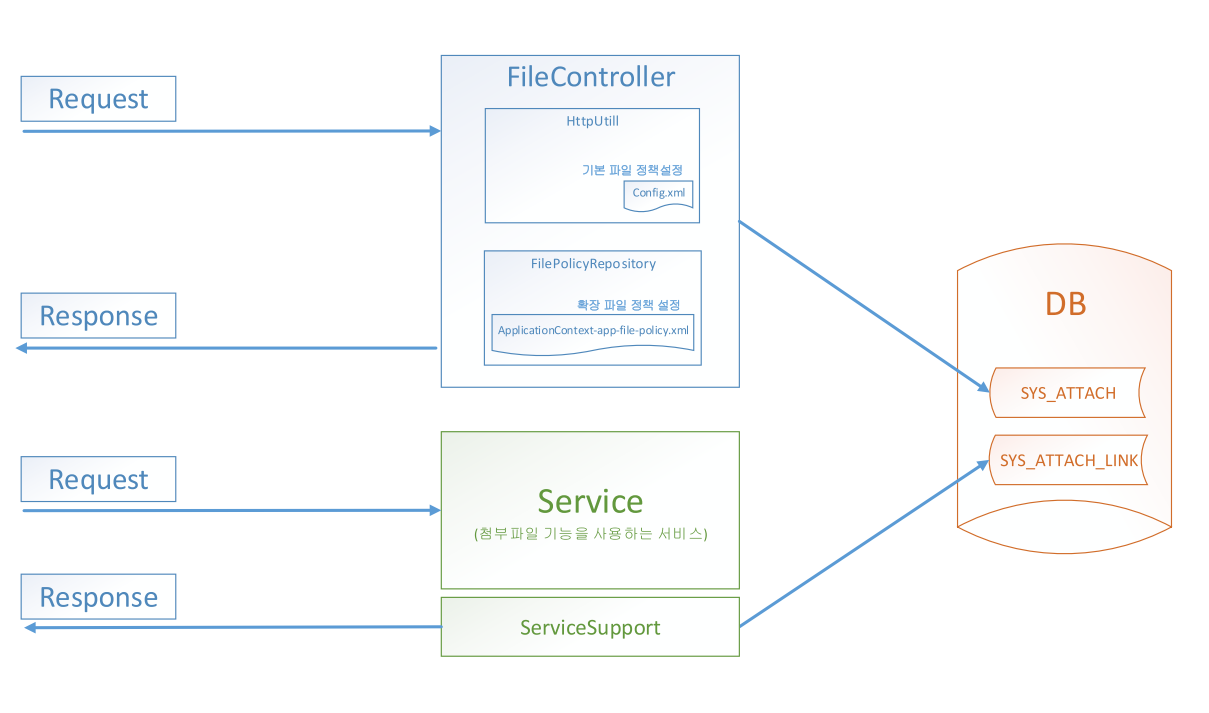
|  |  |
| --- | --- |
| **테이블 명** | **설명** |
| SYS\_ATTACH | 첨부 파일에 대한 정보가 기록되는 테이블 |
| SYS\_ATTACH\_LINK | 각 업무별로 첨부 파일에 대한 참조 값이 저장되는 테이블 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SYS\_ATTACH | | | | |
| Column Name | **Data Type** | **NULLABLE** | **Default** | **Comments** |
| ATTACH\_ID | VARCHAR2(50 BYTE) | N |  | 첨부파일마스터번호 |
| FILE\_NM | VARCHAR2(1000 BYTE) | Y |  | 파일명 |
| FILE\_PATH | VARCHAR2(2000 BYTE) | Y |  | 파일경로 |
| FILE\_SIZE | VARCHAR2(50 BYTE) | Y |  | 사이즈 |
| REG\_DT | TIMESTAMP(6) | N | CURRENT\_TIMESTAMP | 등록일시 |
| CONTENT\_TYPE | VARCHAR2(1000 BYTE) | Y |  | 파일내용유형 |
| ATTACH\_TYPE | VARCHAR2(20 BYTE) | Y |  | 첨부파일유형 |
| ATTACH\_SUB\_TYPE | VARCHAR2(20 BYTE) | Y |  | 첨부파일서브유형 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SYS\_ATTACH\_LINK | | | | |
| Column Name | **Data Type** | **NULLABLE** | **Default** | **Comments** |
| ATTACH\_LINK\_ID | VARCHAR2(50 BYTE) | N |  | 첨부파일링크번호 |
| ATTACH\_ID | VARCHAR2(50 BYTE) | Y |  | 첨부파일마스터번호 |
| BIZ\_TP | VARCHAR2(200 BYTE) | Y |  | 업무구분코드 |
| BIZ\_KEY | VARCHAR2(200 BYTE) | Y |  | 업무키 |

## File-Attach 기능 과정 및 구조

### File-Attach 실행 과정



파일 첨부 기능은 크게 FileController 와 ServiceSupport 나눌 수 있다. FileController 파일 첨부 기능이 동작하는 클래스이고, ServiceSupport는 FileController통해서 업로드한 파일들에 대한 정보와 SYS\_ATTACH\_LINK에 데이터가 맵핑시켜, 해당 파일이 특정 업무의 서비스와 연동되게 만들어 준다.

FileController 내부에 HttpUtill은 Config.xml로부터 설정한 파일 정책들을 가져와서 모든 파일이 기본적으로 업로드가 되면, Config.xml 지정한 기본 물리 저장소 위치에 저장시킨다.

그리고 만약에 확장 파일 정책이 있다면 기본 물리 저장소로부터 파일을 다시 가져와서 확장 정책에 따라서 파일을 확장 정책에 있는 물리 저장 위치로 재배치한다. 그리고 SYS\_ATTACH TABLE의 파일 정보가 입력된다.

1차적으로 파일이 물리 저장소에 업로드가 되고 나서 해당 업무에서, 업로드한 파일을 이용하려면 업무 유형에 따라서 파일 정보를 맵핑 시켜야 한다. 이 때 SeviceSupport는 이 파일 정보와 업무 유형을 맵핑 시키거나, 삭제, 수정에 대한 기능을 맡고 있다. 즉 SYS\_ATTAH\_LINK 테이블에 대한 데이터를 입력, 수정, 삭제, 조회 기능을 맡고 있다.

SYS\_ATTACH\_LINK에는 BIZ\_TP(업무구분코드)과 BIZ\_KEY(업무키)가 저장되는데 AplicationWorkCategory (enum)클래스에서 지정된 enum 상수들이 사용된다. 지정 방법은 다음과 같이 enum 상수에 다음과 같이 선언하면 된다.

/\*\*

\* 해당업무유형명(“업무에대한설명”,업무코드상태값,“해당업무테이블이름”,”해당업무테이블primarykey”);

\* 공지사항

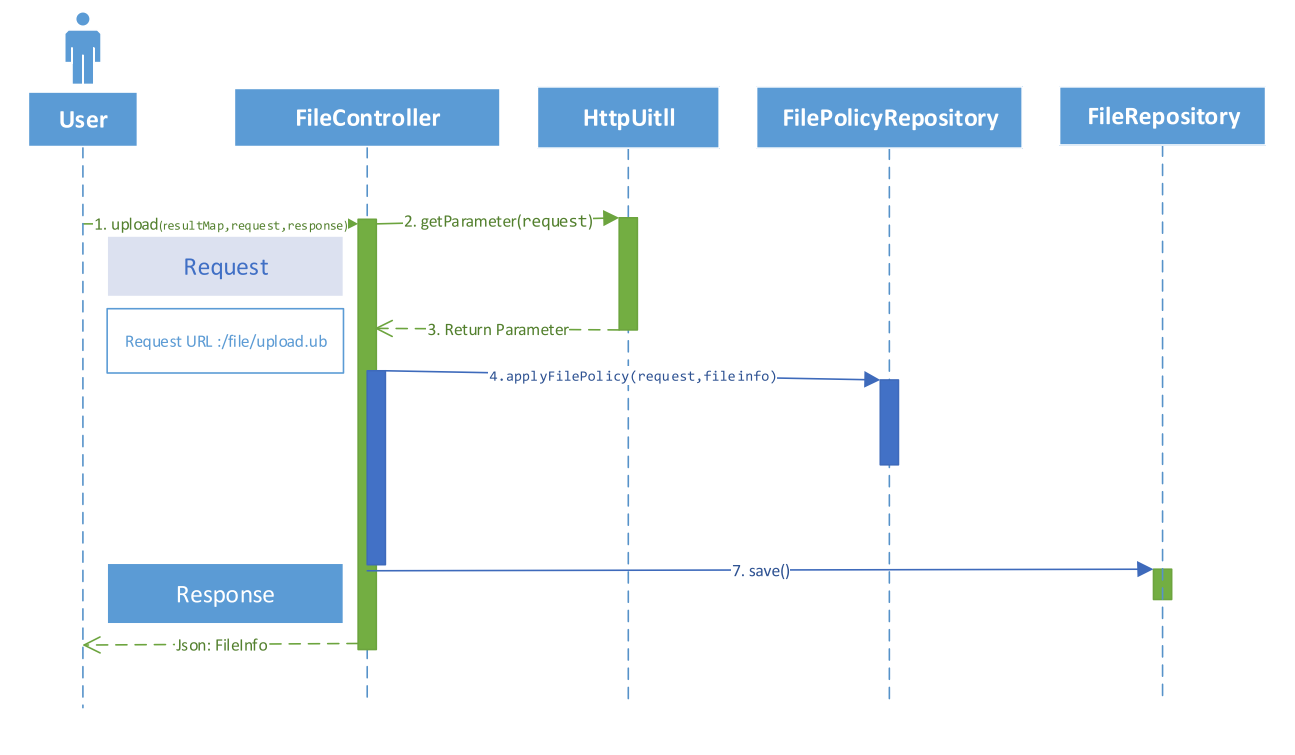
\*/

***NOTICE***("공지사항", Code.***ACTIVITY\_STATUS\_CODE***, "TBS\_NOTICE", "ID\_NOTICE");

다음과 같이 지정한 AplicationWorkCategory 클래스의 enum 상수의 지정된 값으로 NOTICE 는 BIZ\_KEY로 지정되고 ID\_NOTICE의 값은 BIZ\_KEY로 된다.

|  |  |  |
| --- | --- | --- |
| 패키지명 | 파일명 | 역할 |
| com.ubone.framework.web.controller | FileController.class | File Attach에 관한 요청에 대한 Controller |
| com.ubone.framework.util.file | FileRepository.class | FileRepository인터페이스 |
| JdbcFileRepository.class | FileRepository의 구현체 |
| FilePolicyRepository.class | 파일 정책에 관한 저장소 |
| FilePolicy.class | File 정책에 대한 정보를 담는 Bean 객체 |
|  | HttpUtill.class | MutipartRequest를 처리한다. |
|  | AplicationWorkCategory.class | 업무 유형에 대한 Enum 상수를 정의 |

### File-Attach 업로드 시



사용자가 파일 업로드를 버튼을 클릭하여 파일을 열기로 Form 객체에 로딩시키면 form의 onchange 함수가 호출되면서 SwfUpload라이브러리를 통해서 비동기식으로 파일을 Upload한다. 이 때 FileContoller upload()가 호출 되면서 사용자의 Request이 전달된다.

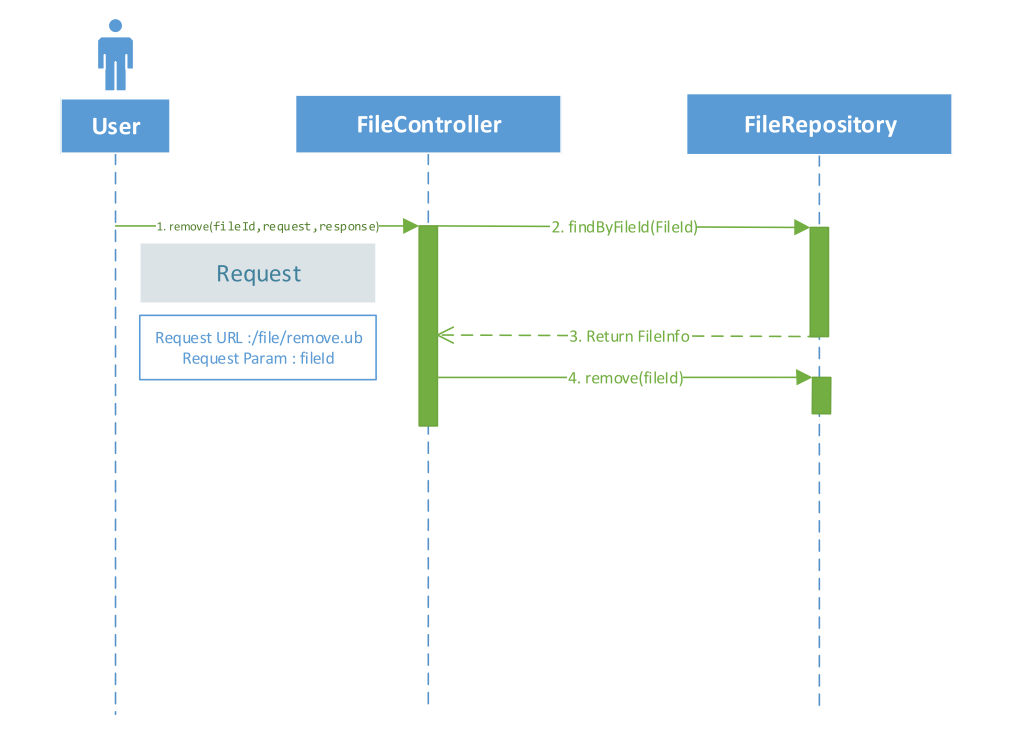
FileController에서 HttpUtill.getParameter()를 호출하면 1차적으로 파일을 Config.xml에 지정한 물리 저장 위치에 임시적으로 저장한다. applyFilePolicy() 호출되면 해당되는 파일 정책을 FilePolicyRepository에서 applicationContext-app-file-policy.xml 지정한 정책들을 가져와서 해당 파일의 이름, 저장 위치에 대한 정보를 변경하고 물리적으로 파일을 다시 저장하고, SYS\_ATTACH Table에 파일에 대한 정보를 저장한다. 최종적으로 파일에 대한 정보를 다시 Json 형태로 넘겨주고 다시 화면에서는 반환 결과를 통해서 화면에 파일이 성공적으로 업로드 된 상태를 보여준다.

* Config.xml 일부분 upload 위치 지정

<!-- 업로드 path -->

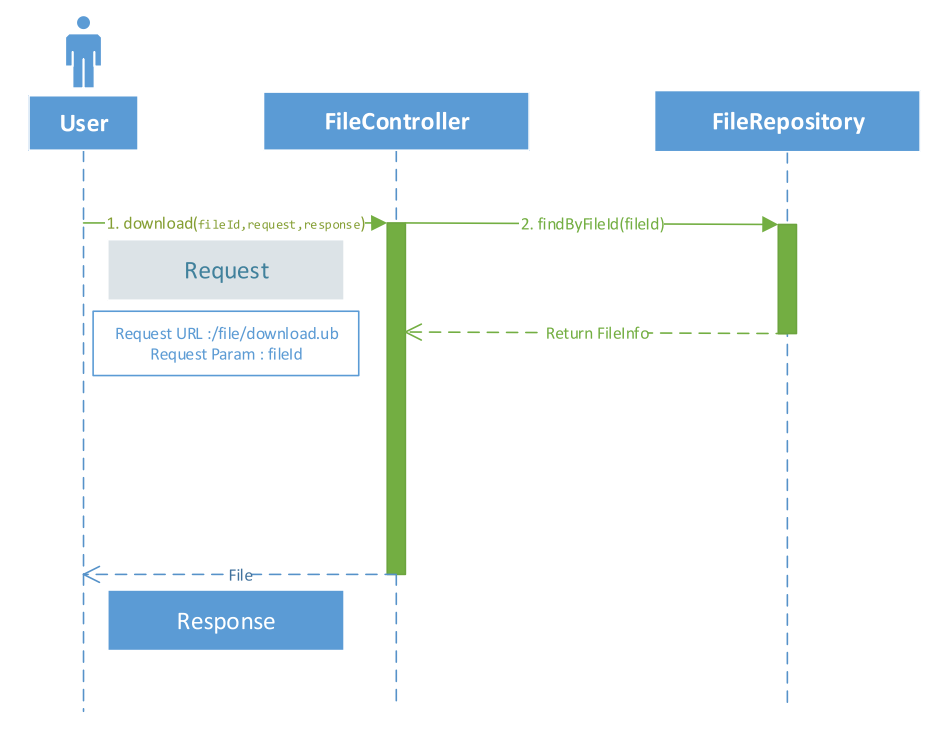
<entry key=*"framework.upload.path"*>c:/upload/ub\_base/files</entry>

### File-Attach 파일 삭제 시

****

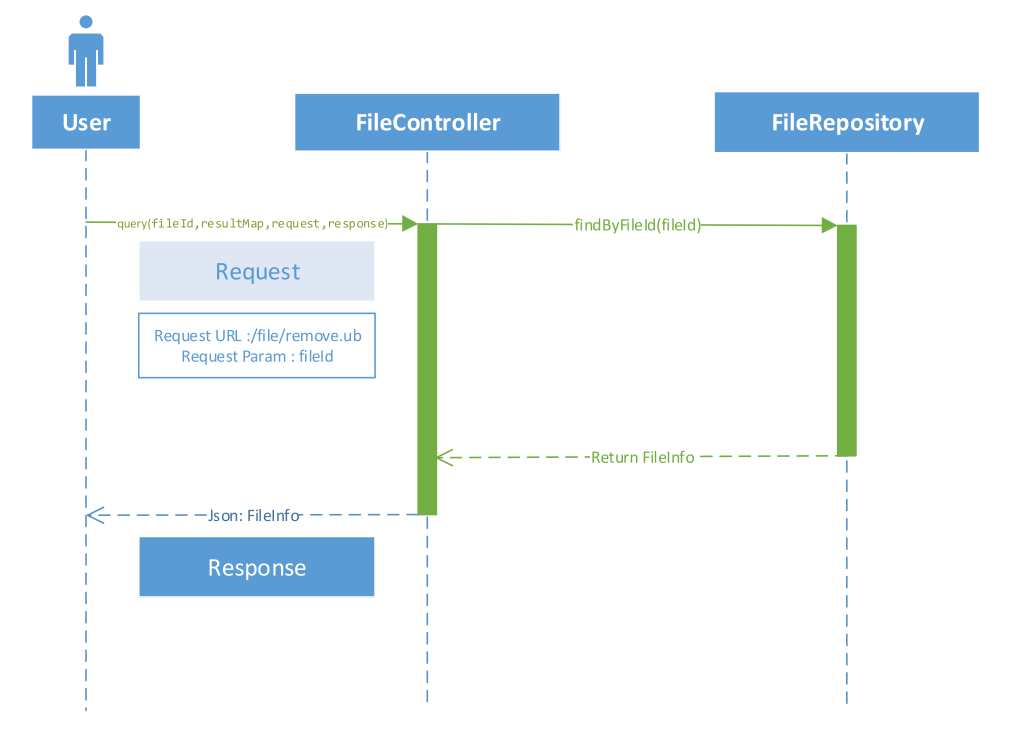
파일 삭제 시에는 해당 파일의 FILE\_ID를 파라미터로 보내 FileRepositoy에서 해당 파일에 대한 정보를 가져와서 해당 파일을 먼저 파일 정보를 DB에서 삭제한 후에, 물리적으로도 삭제한다.

### File-Attach 파일 다운로드 시



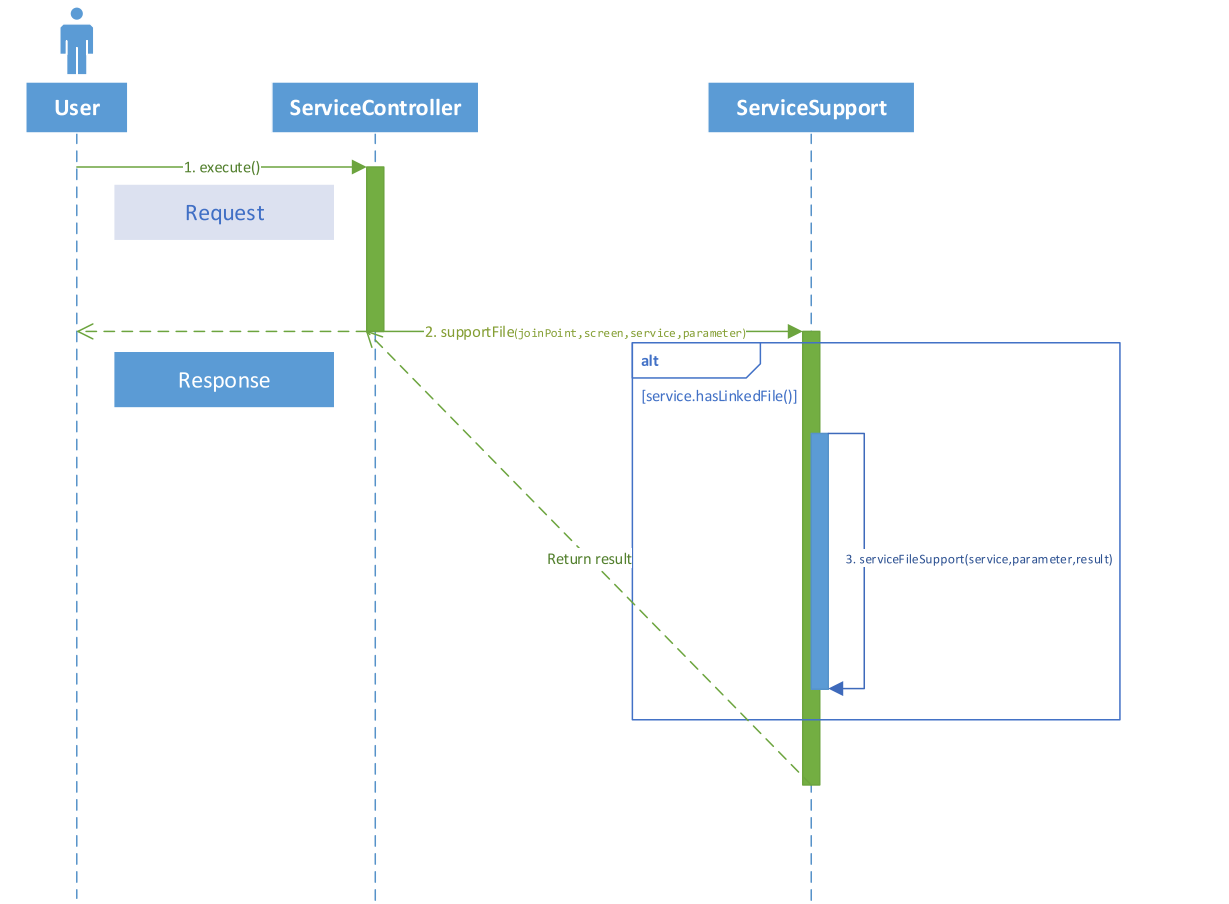
파일 다운로드 시에 FILE\_ID를 보내서 해당 파일에 대한 정보를 가져와서 FILE 객체를 가져와서 Response 객체에 담아서 해당 파일을 전송한다.

### File-Attach 파일 정보 조회 시



파일 다운로드 시에 FILE\_ID를 보내서 해당 파일에 대한 정보를 JSON 형식으로 리턴 받는다.

### ServiceSupport 첨부 파일 지원 기능



APMD에 등록된 서비스를 호출하면 AOP로 내부적으로 ServiceSupport에 있는 @Around Advice형식으로 다음과 같이 구현되어 있다.

@Around("executionService(screen, service, parameter)")

**public** Object supportFile(ProceedingJoinPoint joinPoint, Screen screen, Service service, Parameter parameter) **throws** Throwable {

Result result = (Result) joinPoint.proceed();

**if**(service.hasLinkedFile()) {

serviceFileSupport(service, parameter, result);

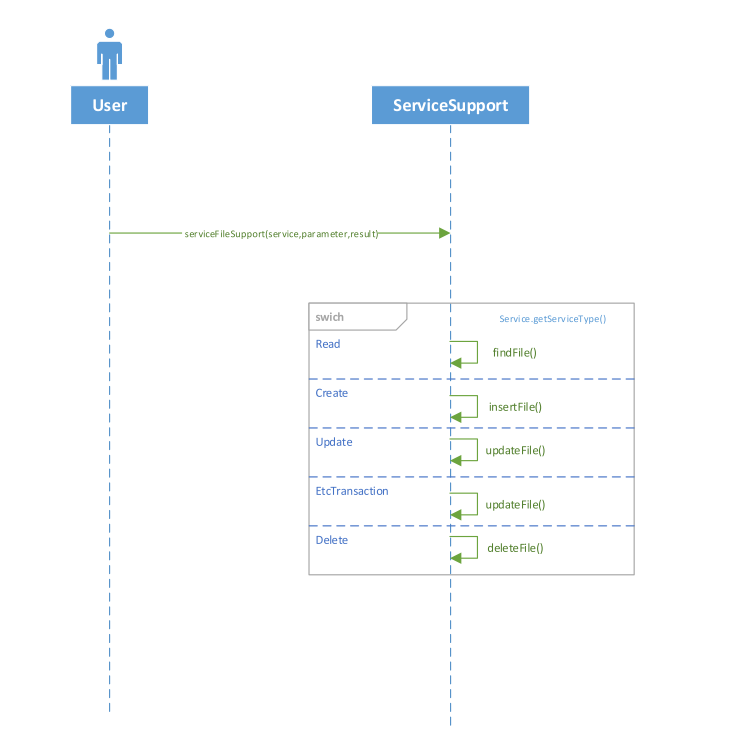
}

**return** result;

}

service.hasLinkedFile() 은 첨부 파일 기능 사용 여부를 체크해서 serviceFileSupport()을 실행한다.

그리고 서비스 유형대로 분기하여 첨부 파일 관련 기능을 실행한다.



예를 들면 게시판에서 글을 등록하는데 글 등록 서비스는 첨부파일기능도 사용한다. 글과 첨부파일이 등록되면, 해당 업무 유형에 따라서 등록이므로 Create 서비스 유형으로 insertFile()를 호출하여 SYS\_ATTACH\_LINK 테이블에 해당 BIZ\_TYPE(업무유형) , BIZ\_KEY(글번호 등)이 INSERT된다. 나머지 설명 기능도 비슷한 방식으로 동작한다.

# File Attach 기본적인 사용

## File Attach 등록 화면 예제

### File Attach 등록 화면 Meta-Studio에서 ATTACH-FILE Control 추가 및 등록

다음과 같이 AplicationWorkCategory.class enum 상수를 추가한다.

/\*\*

\* enum 상수 선언 시

\* PrimaryKey가 단독인 경우

\* 업무유형명(“업무설명”, 활성화상태, “테이블이름”, “PrimaryKeyName”)

\* PrimaryKey가 복수인 경우

\* 업무유형명(“업무설명”,활성화상태,“테이블이름”,“PrimaryKeyName1”,”PrimaryKeyName2”,”PrimaryKeyName3”)

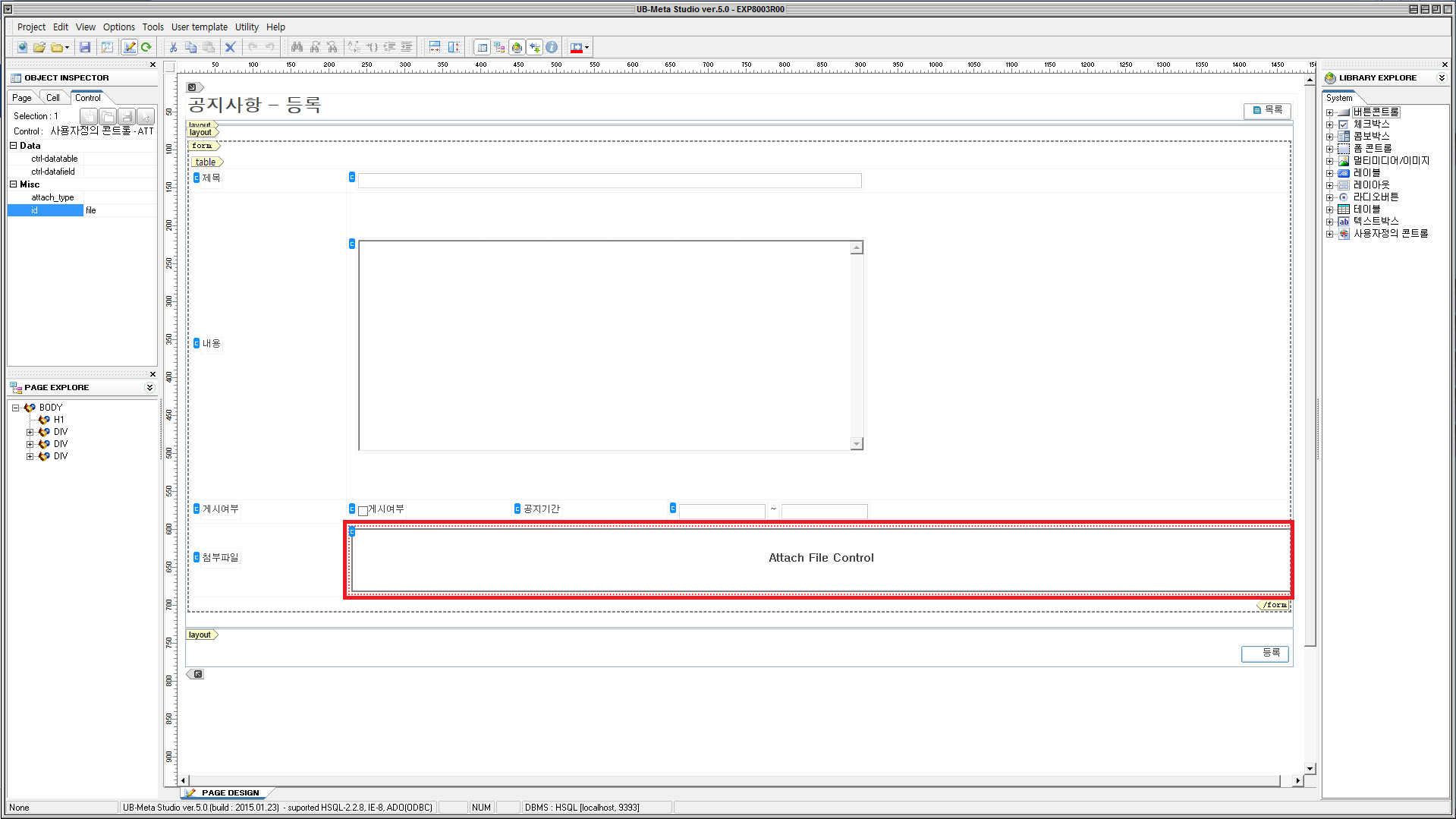
\*

\*/

***NOTICE***("공지사항", Code.***ACTIVITY\_STATUS\_CODE***, "TBS\_NOTICE", "ID\_NOTICE")

다음은 Meta-Studio에서 Attach–File Control 를 추가한다.

사용자 정의 컨트롤 > 첨부파일 – 첨부 파일 컨트롤



### File Attach 등록 화면 File-control-id 등록

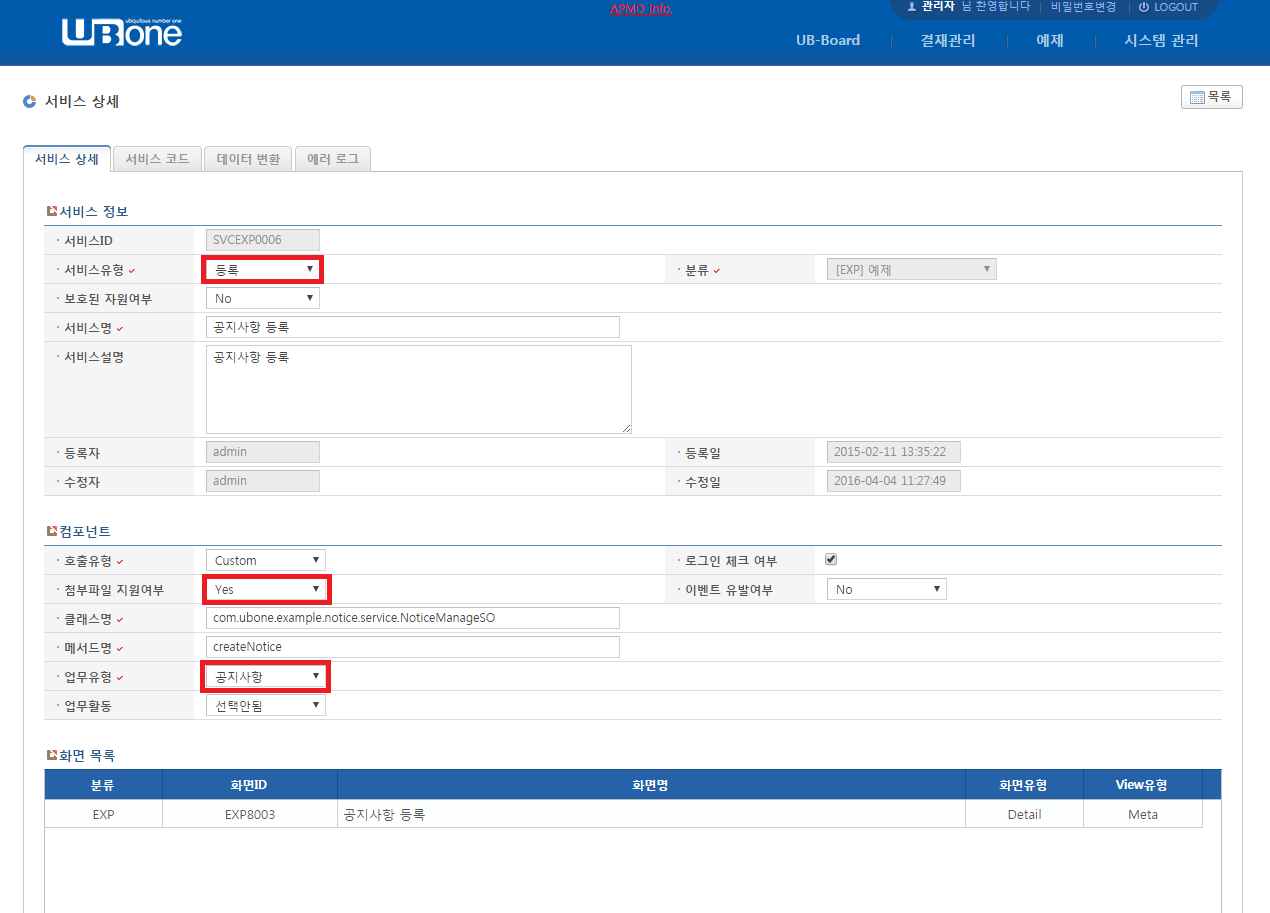
id 에 file로 등록한다.(id는 파일 정책에 따라서 지정해줘야 한다.)



### APMD 서비스 설정

서비스 설정 메뉴 : 시스템관리 > 프로그램 관리(APMD)> 서비스관리

등록 화면에 지정된 등록 서비스에 첨부파일 기능 지원 여부를 YES 바꾼다. 그리고 업무 유형을 공지사항을 선택한다.



→ 첨부파일 지원 여부와 YES 체크하고 서비스 유형도 등록으로 지정해준다. 서비스 유형은 조회, 등록, 수정, 삭제에 따라서 파일 첨부에 대한 서비스 지원 기능이 동작하므로 해당 서비스의 유형에 맞게 등록 해야 한다.

그리고 업무 유형은 SYS\_ATTACH\_LINK Table에 파일에 대한 정보가 Insert 될 때 BIZ\_TP 의 값으로 지정되므로 서비스 유형, 첨부파일 지원여부, 업무 유형은 반드시 정확히 지정해야 정상적으로 파일 첨부 기능이 동작 한다 .

### Jsp 파일에 uxl.ext.Upload 추가

uxl.onLoad() 안에서 uxl.ext.Upload(‘file’)을 다음과 선언한다.

<script type=*"text/javascript"*>

uxl.onLoad(**function**(){

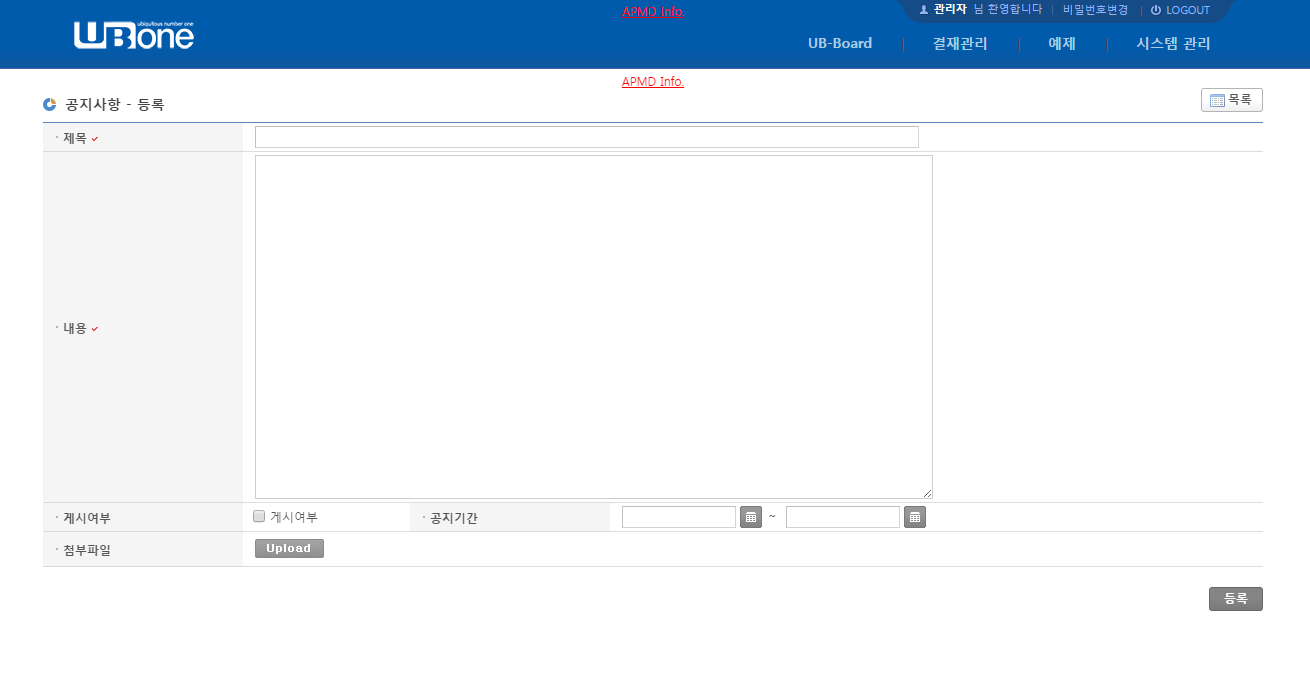
uxl.ext.Upload('file');

});

</script>

→ Attach-File-Control의 ID 는 파일 정책을 구분하는 기준인 AttachType의 값이 된다. 그러므로 해당 파일에 특정 파일 정책이 따라서 ID를 모두 통일해서 사용해야 한다.

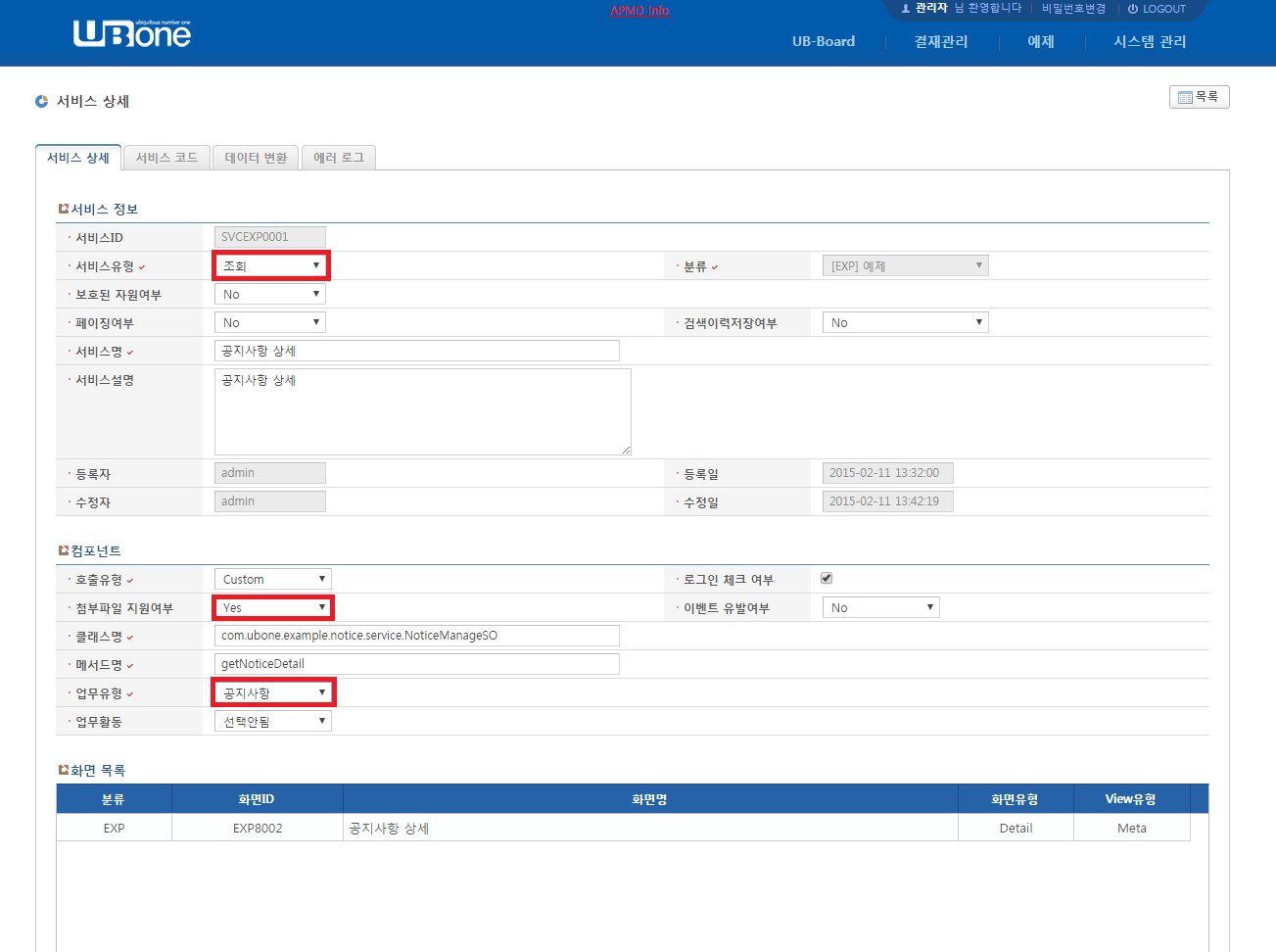
다음과 같이 선언하게 되면 Upload 버튼이 생성된다.

****

## File Attach 상세화면 예제

### APMD 서비스 설정(상세화면)

APMD에서 서비스 등록 할 때 서비스 유형, 첨부파일 지원, 업무 유형을 지정한다.



### uxl.ext.upload Option 설정 바꾸기

상세 화면일 경우에는 첨부한 파일을 수정하지 못하게 바꾸어야 한다. 이 경우에는 다음과 같이 옵션을 지정하여 사용하면 된다. 기본적으로 readOnly 속성은 false로 되어 있으므로 파일을 수정 가능하게 하고 싶다면 따로 옵션을 지정하지 않아도 된다.

<script type=*"text/javascript"*>

uxl.onLoad(**function**(){

// 파일 수정 모드 활성 여부

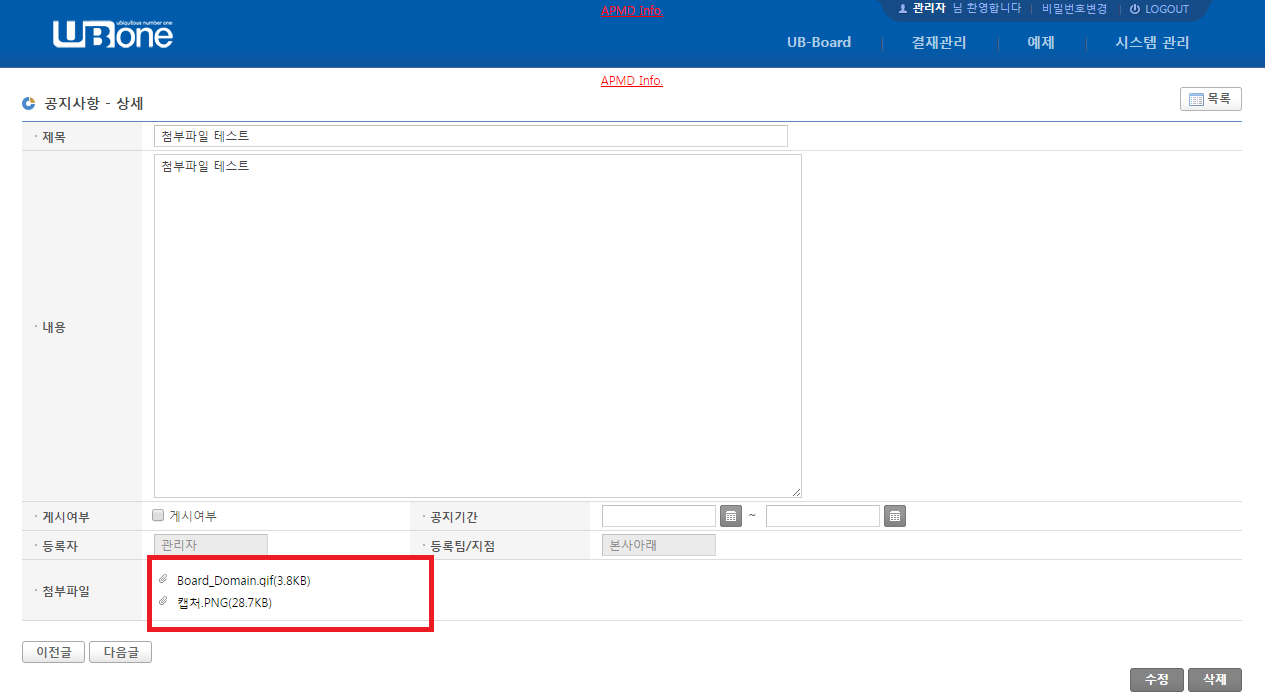
**var** file\_Options = { readOnly : **true** };

uxl.ext.Upload('file',file\_Options);

});

</script>

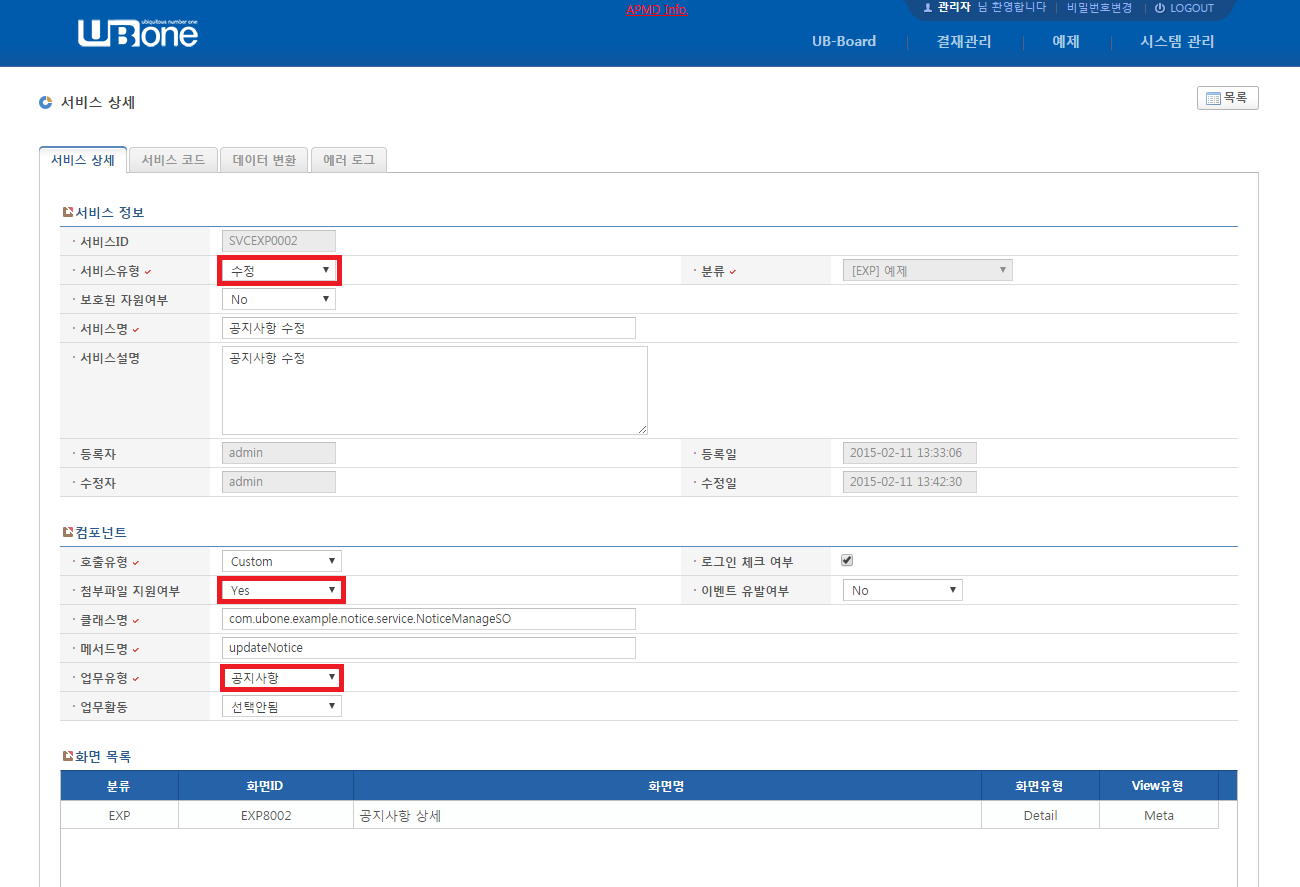
등록화면에서 설정 없이 작성한 것과 달리 삭제버튼은 사라지고 파일 다운로드만 가능해졌다.



## File Attach 수정화면 예제

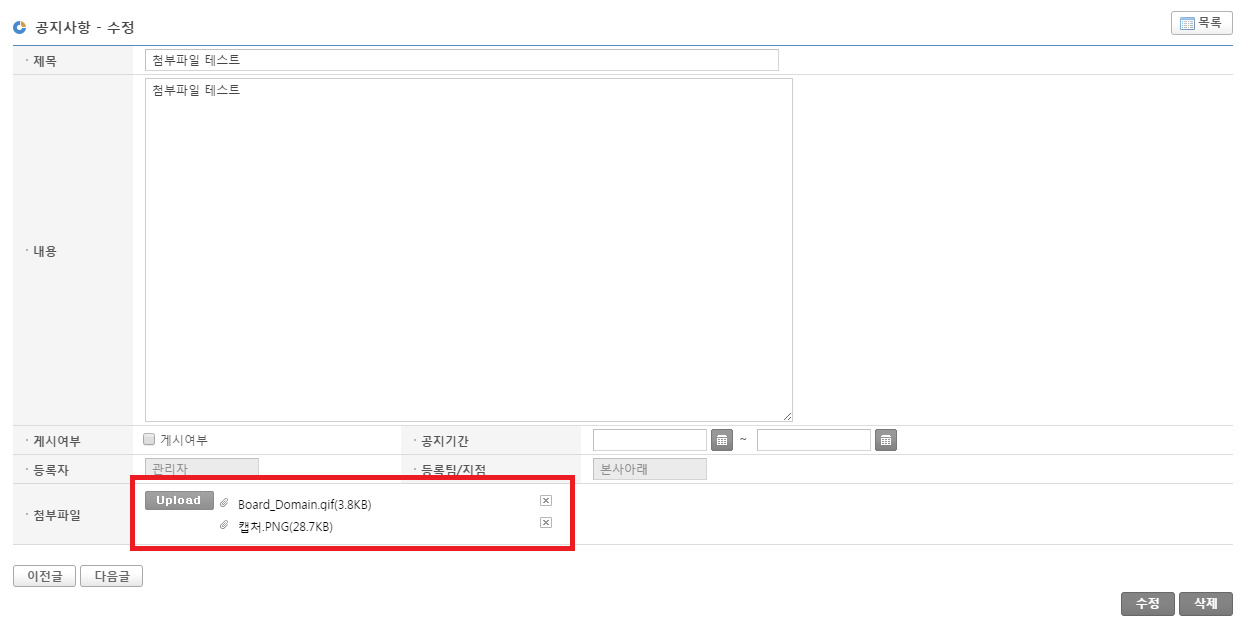
### APMD 서비스 설정(수정 서비스)

서비스 유형은 수정, 첨부파일 지원여부를 YES, 업무 유형은 공지사항으로 체크한다.



### File Attach 수정화면 예제

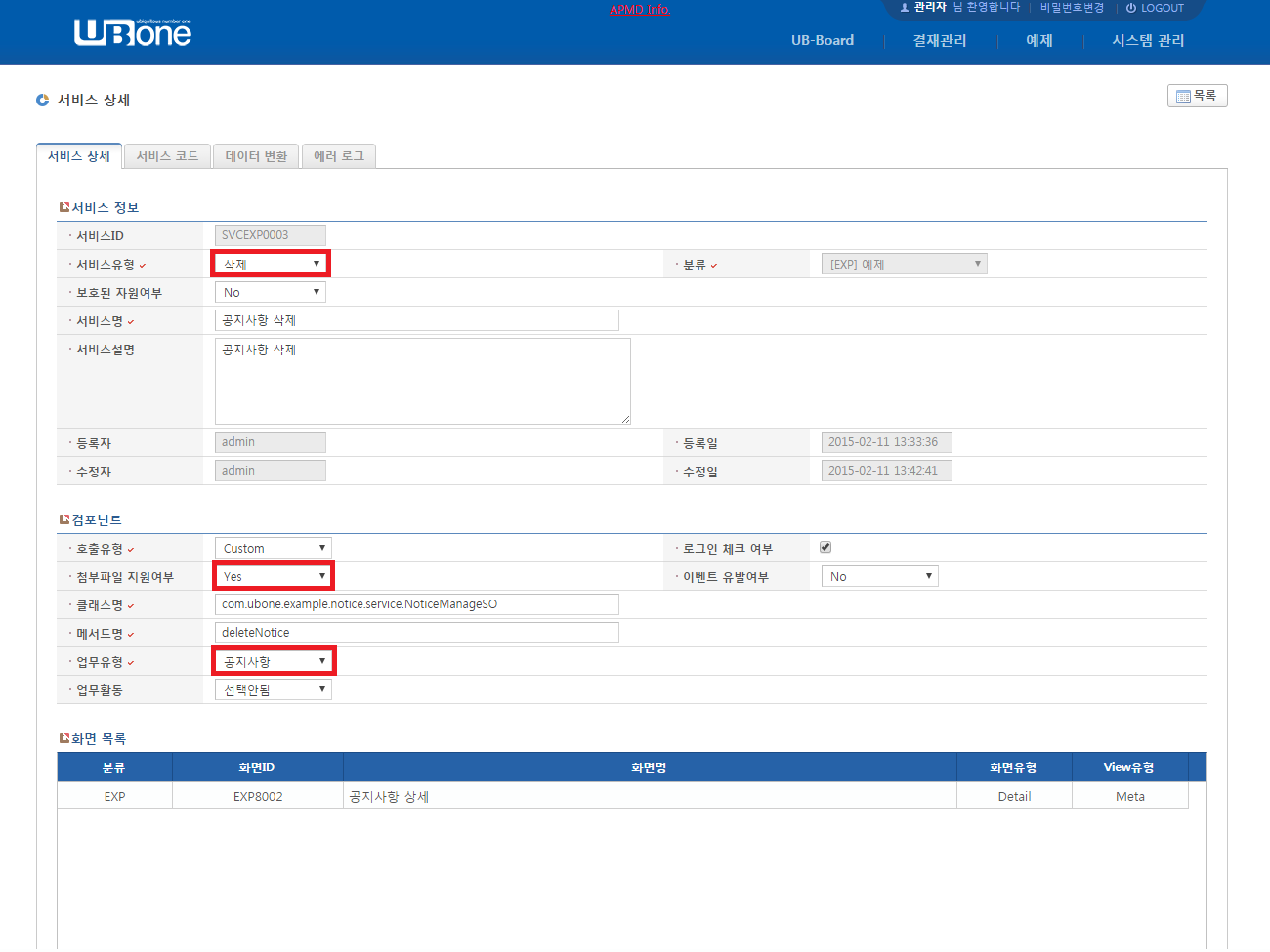
readOnly 옵션은 Default 값이 false 이므로 따로 지정하지 않아도 된다.



## File Attach 삭제화면 예제

### APMD 서비스 설정(삭제 서비스)

서비스 유형은 수정, 첨부파일 지원여부를 YES, 업무 유형은 공지사항으로 체크한다.



# File Attach uxl.ext.Upload부가 옵션

## uxl.ext.Upload옵션

아래 코드와 같이 uxl.ext.Upload() 에 다양한 옵션을 설정하여 사용가능 하다.

<script type=*"text/javascript"*>

uxl.onLoad(**function**(){

**var** file\_Options = {

file\_size\_limit : 1024 \* 100 // 파일 크기 제한 지정

, allowFileType : "\*.jpg;\*.jpeg;\*.pdf;\*.png" // 업로드 가능 확장자 지정

, file\_upload\_limit : 10 // 파일 업로드 수 제한

, file\_queue\_limit : 10 // 동시 업로드 갯수 제한

, uploadSuccessAfter : **function**(){ // 업로드 성공후 CallBack 함수 지정

uxl.showMessage("파일 업로드 성공 !!");

}

, removeAfter : **function**(){ // 파일 삭제 이후 CallBack 함수 지정

uxl.showMessage("파일 삭제 성공 !!");

}

, readOnly : **false** // 파일 수정 모드 활성 여부

};

uxl.ext.Upload('file',file\_Options);

});

</script>

→ 파일 업로드 사이즈, 허용 가능 확장자, 파일 업로드 수 제한 설정은 Config.xml과 applicationContext-app-file-policy.xml 에서도 지정 하므로, 추가 옵션을 사용할 때는 xml에서 설정된 파일 정책에 맞추어 사용하면 된다.

참고파일 확장자 지정 방법

allowFileType : "\*.\*" -> 모든 파일 유형 확장자 가능

allowFileType : "\*.jpg;\*.jpeg;\*.pdf;\*.png" -> 개별적으로도 지정가능

# File Attach 관련 xml 설정

## Config.xml Upload 설정

Config.xml 에서 기본적으로 따라서 파일 기본 파일 정책을 설정하여 저장되는 Path를 지정할 수 있다.

그리고 추가적으로 상세한 설정은 파일 정책을 설정하는 applicationContext-app-file-policy.xml 파일에서 지정 가능 하다.

<!--

====================================================================

file upload 관련 설정

====================================================================

-->

<!-- 업로드 path -->

<entry key=*"framework.upload.path"*>c:/upload/ub\_base/files</entry>

→ 기본적으로 업로드 한 모든 파일이 저장되는 기본 Path로 지정된다.

<!-- 파일 업로드 건수 제한 -->

<entry key=*"framework.upload.count.limit"*>10</entry>

→ 모든 업로드에 대한 건수를 제한한다.

<!-- 파일 사이즈 제한(단위 MB) -->

<entry key=*"framework.upload.size.limit"*>10</entry>

→ 모든 업로드 파일에 대한 사이즈를 제한한다.

<!--

업로드 확장자 제한 유형 (default : WHITE)

\* BLACK - Black List : 명기된 확장자만 업로드 할 수 없는 방식

\* WHITE - White List : 명기된 확장자만 업로드 할 수 있는 방식

-->

<entry key=*"framework.upload.ext.limit.type"*>WHITE</entry>

<!-- 업로드 확장자 제한 리스트 -->

<entry key=*"framework.upload.ext.limit.list"*>hwp;doc;xls;ppt;pptx;png;gif;jpg;jpeg;txt;</entry>

## applicationContext-app-file-policy.xml 설정

기본 정책 외 추가적인 파일 정책 설정하기 위해서는FilePolicyRepostory의 Bean에

List<FilePolicy> extendFilePolicies로 형태로 저장되므로 다음과 같이 선언한다.

<bean id=*"filePolicyRepository"* class=*"com.ubone.framework.util.file.FilePolicyRepository"*>

<property name=*"extendFilePolicies"*>

<list>

<bean class=*"com.ubone.framework.util.file.FilePolicy"*></bean>

</list>

</property>

</bean>

FilePoily Bean 경우는 다음과 같이 설정 할 수있다.

<bean class=*"com.ubone.framework.util.file.FilePolicy"*>

<!-- 적용될 AttachType리스트 Ex ) thumbnail-->

<property name=*"attachTypes"*>

<list>

<value>AttachType명</value>

</list>

</property>

<!-- 파일 저장 위치 Ex) value=”c:/base/upload” -->

<property name=*"baseDir"*  value=*"저장경로"* />

<!-- 파일 중간 경로 생성 규칙 Ex) value=“1” -->

<!--0-None(중간경로 없음),1-YYYY, 2-YYYY/MM, 3-YYYY/MM/DD, 4-YYYYMM, 5-YYYYMMDD-->

<property name=*"pathRule"* value=*"중간경로생성규칙번호"* />

<!-- 파일명 생성 규칙(디스크에 저장되는 파일명 형식) Ex) value=”2” -->

<!-- 0-랜덤파일명.확장자, 1-랜덤파일명, 2-파일명.확장자, 3-파일명 -->

<property name=*"nameRule"* value=*"파일명생성규칙번호"*/>

<!--

업로드 확장자 제한 유형 (default : WHITE)

\* BLACK - Black List : 명기된 확장자만 업로드 할 수 없는 방식

\* WHITE - White List : 명기된 확장자만 업로드 할 수 있는 방식

-->

<property name=*"secureType"* value=*"WHITE"* />

<!-- 업로드 파일확장자 제한 리스트(대소문자 구분 없음) -->

<property name=*"limitFileExts"*>

<list>

<!-- Hangul File List -->

<value>hwp</value>

<!-- MS-Office File List -->

<value>doc;docx;xls;xlsx;ppt;pptx</value>

<!-- Image File List -->

<value>bmp;png;gif;jpg;jpeg</value>

<!-- Document File List -->

<value>txt;pdf</value>

</list>

</property>

</bean>

# File Attach Annotation

## @FileSupport Annotation 사용

-> 첨부파일 지원이 필요할 경우 Custom Service Method 단위로 FileSupport Annotation 을 선언한다.

@FileSupport : 해당 서비스 실행 시에 첨부파일에 대한 지원이 필요할 경우에 선언한다.

* primaryKeyFieldNames : 업무 도메인의 Primary Key 명을 설정한다.
* 첨부파일 조회/ 수정 / 삭제 시에는 메소드의 Argument 에 설정된 Primary Key 명에 해당되는 입력 값이 포함 되어야한다.
* 첨부파일 저장 시에는 Primary Key에 해당되는 파라미터가 포함되어야 한다.
* methodType : 파일 지원에 대한 상세한 유형을 설정한다.

(Create ,Read ,Update ,Read )

@KeyParameter : 입력 파마티터로 특정 항목을 명시적으로 사용하고 싶을 경우 (특정 업무의 상세조회 같이 해당 업무의 Primary Key만을 넘겨 받아 사용하는 경우) 선언한다.

* Name : 명시적으로 받아 사용할 항목 명을 설정한다.
* Required : 설정한 항목의 필수 여부를 설정한다.(default : true)
* 필수 여부가 True 일 때 해당 값이 null 이거나 빈값 일 경우 ServiceExeption을 던진다.

@FileSupport(primaryKeyFieldNames="ID\_NOTICE", methodType=FileMethodType.***READ***)

**public** Result getNoticeList(@KeyParameter(name="ID\_NOTICE", required=**true**)

Parameter parameter){

// 구현 내용 생략 …

**return** rs;

}

/\*\*

\* **@FileSupport** 어노테이션 사용 예시

\*

\*/

// 단순 조회 일 경우

@FileSupport(primaryKeyFieldNames="PrimaryKey 이름 ", methodType=FileMethodType.***READ***)

**public** Result List(@KeyParameter(name="PrimaryKey 이름 ", required=**true**) Parameter parameter){

// 구현 내용 생략 …

**return** rs;

}

// 등록일 경우

@FileSupport(primaryKeyFieldNames="PrimaryKey 이름 ", methodType=FileMethodType.***CREATE***)

**public** Result insert(@KeyParameter(name="PrimaryKey 이름 ", required=**true**) Parameter parameter){

// 구현 내용 생략 …

**return** rs;

}

// 수정일 경우

@FileSupport(primaryKeyFieldNames="PrimaryKey 이름 ", methodType=FileMethodType.***UPDATE***)

**public** Result update(@KeyParameter(name="PrimaryKey 이름 ", required=**true**) Parameter parameter){

// 구현 내용 생략 …

**return** rs;

}

// 삭제일 경우

@FileSupport(primaryKeyFieldNames="PrimaryKey 이름 ", methodType=FileMethodType.***DELETE***)

**public** Result delete(@KeyParameter(name="PrimaryKey 이름 ", required=**true**) Parameter parameter){

// 구현 내용 생략 …

**return** rs;

}