

**eXBuilder6**

# 교육교재

(심화교육)



**TOMATO SYSTEM**

**[문서 이력]**

[illegible]

## [목 차]

1.	개요 .....	6
2.	프로젝트 표준 .....	6
2.1.	스튜디오 설정 .....	6
2.1.1.	사용자 정의 속성 .....	6
2.1.2.	스크린 .....	7
2.1.3.	ID 정책 .....	8
2.1.4.	UI 템플릿 .....	8
2.1.5.	코드 템플릿 .....	9
2.2.	환경 설정 .....	11
2.2.1.	env.json .....	11
2.2.2.	defaults.js .....	13
2.2.3.	exbuilder.json .....	13
2.2.4.	디플로이 환경 파일 위치 .....	16
3.	앱 .....	16
3.1.	앱 .....	16
3.2.	앱 인스턴스 .....	17
3.3.	앱 속성 출판 .....	18
3.4.	앱 이벤트 출판 .....	19
3.5.	앱 함수 출판 .....	19
4.	임베디드 앱 .....	20
4.1.	임베디드 앱 인스턴스 접근 .....	20
4.2.	임베디드 앱에서 부모 앱 인스턴스 접근 .....	21
4.3.	최상위 앱 인스턴스 접근 .....	21
4.4.	임베디드 앱 강제 실행 .....	21
5.	UDC .....	21
5.1.	그리드 내에서 사용자 정의 컨트롤(UDC) 사용 방법 .....	22
5.2.	개발 유의점 .....	22
5.3.	임베디드 앱과 UDC 의 차이점 .....	22
6.	외부 연동 .....	23
6.1.	임베디드 페이지 .....	23
6.1.1.	임베디드된 페이지의 함수 호출 및 파라미터 전송하는 방법 .....	23
6.2.	웹 컨트롤 .....	24
6.2.1.	외부 라이브러리 연동 방법 .....	26
7.	UI 컨트롤 .....	28
7.1.	렌더링(그리기) 과정 .....	28
7.2.	Virtual DOM .....	29
7.3.	바인딩 .....	30
7.3.1.	바인딩 유형 .....	30

7.3.2.	표현식 엔진 .....	34
7.4.	스타일링 .....	37
8.	레이아웃 .....	38
8.1.	레이아웃 유형 .....	38
8.2.	컨스트레인트 지정 .....	39
8.2.1.	XY 레이아웃 .....	39
8.2.2.	반응형 XY 레이아웃 .....	41
8.2.3.	폼 레이아웃 .....	42
8.2.4.	버티컬 레이아웃 .....	43
8.2.5.	플로우 레이아웃 .....	44
9.	이벤트 .....	45
9.1.	이벤트 순서 .....	45
9.1.1.	컨트롤의 기본 동작 차단 .....	45
9.2.	이벤트 버스 및 필터 .....	46
9.3.	알림 .....	48
10.	서브미션 .....	49
10.1.	서브미션 이벤트 순서 .....	49
10.2.	서브미션 속성 설명 .....	51
10.3.	데이터 형식 변환 .....	52
10.4.	엑셀 export .....	54
10.4.1.	엑셀 & PDF export .....	54
10.5.	대용량 데이터 .....	56
10.6.	파일업로드/다운로드 .....	57
10.6.1.	파일업로드 .....	57
10.6.2.	파일 다운로드 .....	60
10.7.	UI Adaptor(with springframework) .....	62
11.	모듈 .....	65
11.1.	모듈 출판 방식 .....	65
11.1.1.	글로벌 출판 .....	65
11.1.2.	모듈 멤버 출판 .....	66
11.2.	세만틱 콘텐츠 어시스트 .....	66
11.2.1.	세만틱스 타입 추론 .....	67
11.3.	모듈 의존성 .....	68
11.3.1.	런타임 의존성 .....	68
11.3.2.	정적 의존성 .....	69
11.4.	모듈 작성 가이드 라인 .....	69
12.	CI 연동 .....	70
12.1.	eXBuilder6 도구 내보내기 .....	70
12.2.	컴파일 .....	71



## 1. 개요

본 문서에서는 기본교육을 이수하였다는 전제하에 작성되었으며, eXBuilder6 를 통해 개발하는 개발자들에게 기본교육에서는 다루지 않았던 깊이 있는 내용을 다루는 문서입니다. 궁극적으로 문서를 통하여 eXBuilder6 에 대한 이해도, 개발생산성을 높이는 것이 목적이며, **프로젝트 공통 담당자 및 TA 그룹을 대상으로** 진행됩니다.

## 2. 프로젝트 표준

### 2.1. 스튜디오 설정

eXBuilder6 에서는 프로젝트 표준 정의를 위한 여러가지 스튜디오 설정이 있습니다.

#### 2.1.1. 사용자 정의 속성

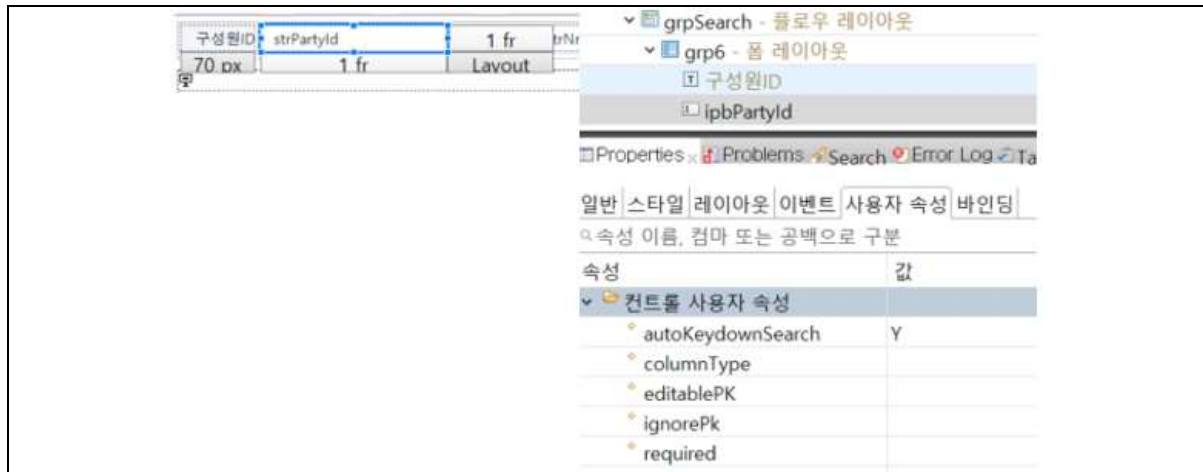
사용자 정의 속성은 모든 컨트롤들에 기본적으로 제공되는 속성 이외에 프로젝트별 응용 가능한 컨트롤별 속성입니다.

공통담당자는 컨트롤 유형별로 기대되는 사용자 정의 속성 집합을 사전에 정의하고 팀원들에게 공유할 수 있습니다. 아래는 인풋 박스 컨트롤에 사용자 정의 속성을 추가하고 디자인 뷰에 인풋 박스 컨트롤을 배치하고 사용자 속성 탭을 확인한 이미지입니다.

프로젝트 전용 설정 구성을 체크하면 .settings 폴더에 custom-properties.xmi 파일이 생성되며 추가된 내용이 작성됩니다.



[Properties > eXBuilder6 > 프로젝트 표준 > 사용자 정의 속성]

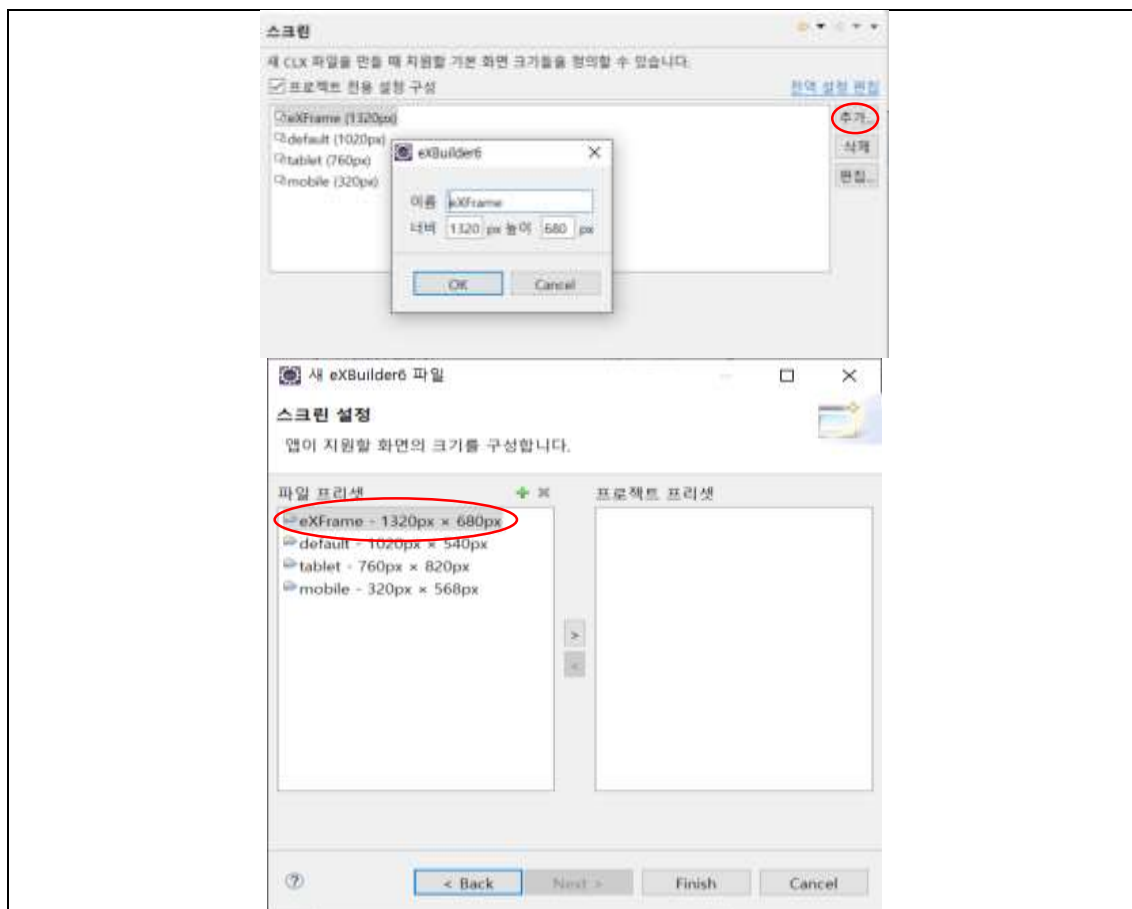


### 2.1.2. 스크린

eXBuilder6 앱이 지원할 스크린의 구성과 크기를 설정할 수 있습니다.

기획 및 디자인 결정 단계 후 지원되는 **최소해상도를 고려하여 스크린 사이즈를 결정**하는 것이 중요합니다. 프로젝트 별로 상이하지만 화면에 스크롤이 생기지 않는 최소 크기를 지정해야 합니다.

프로젝트 전용 설정 구성을 체크하면 .settings 폴더에 screens.xml 파일이 생성되며 추가된 내용이 작성됩니다.



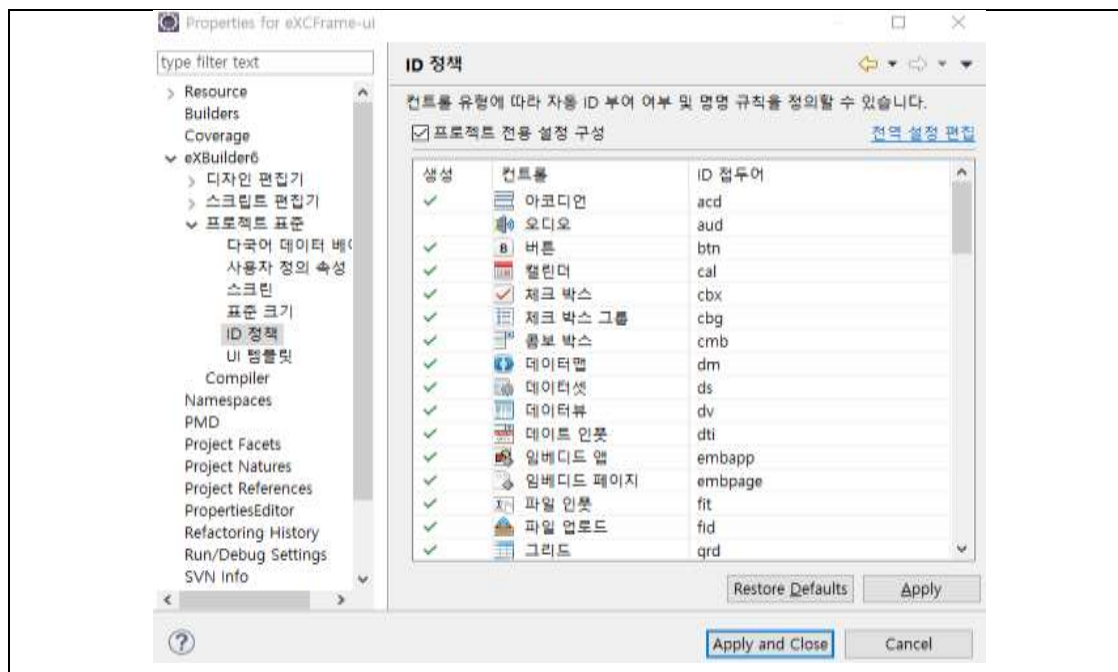
[Properties>eXBuilder6>프로젝트 표준>스크린]

우측 이미지와 같이 프로젝트의 스크린을 설정 후 **새 eXBuilder6 파일을 생성**할 경우에 구성된 스크린이 자동으로 추가됩니다.

### 2.1.3. ID 정책

디자인 편집기에 컨트롤을 배치 할 때 **컨트롤 ID의 접두어(prefix)**를 부여할 수 있습니다. App.lookup() API 를 사용하여 ID 를 부여한 컨트롤 객체를 가져오기 위해서 ID 정책을 설정합니다.

프로젝트 표준 네이밍 룰 기준으로 각 컨트롤의 ID 접두어를 작성하여 사용하시면 됩니다. 프로젝트 전용 설정 구성을 체크하면 .settings 폴더에 id-policy.xml 파일이 생성되며 추가된 내용이 작성됩니다.



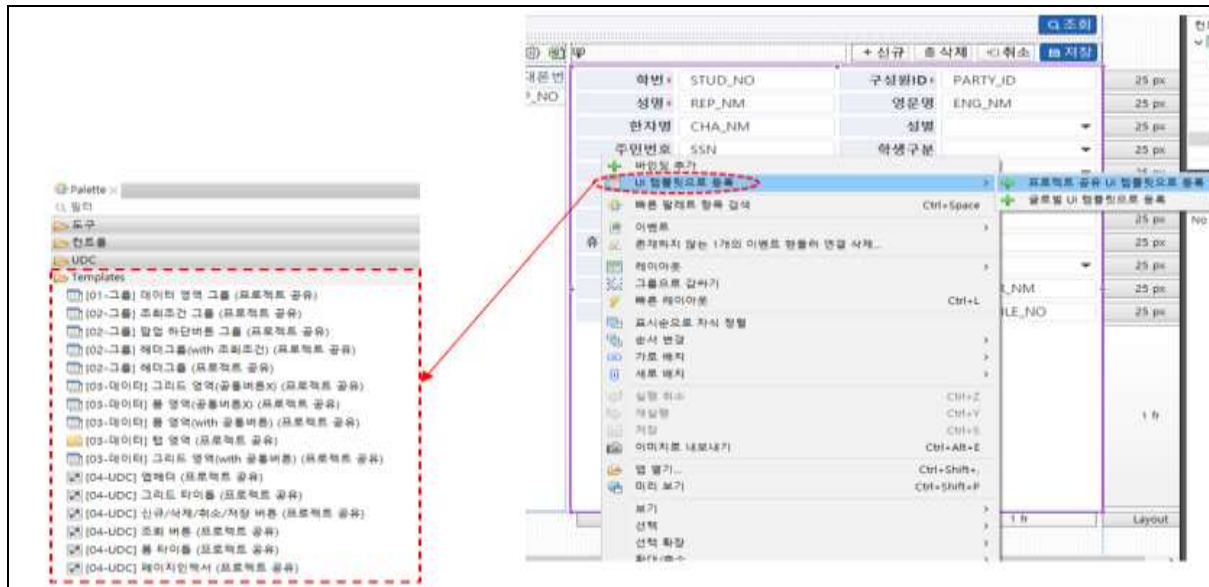
[Properties > eXBuilder6 > 프로젝트 표준 > ID 정책]

### 2.1.4. UI 템플릿

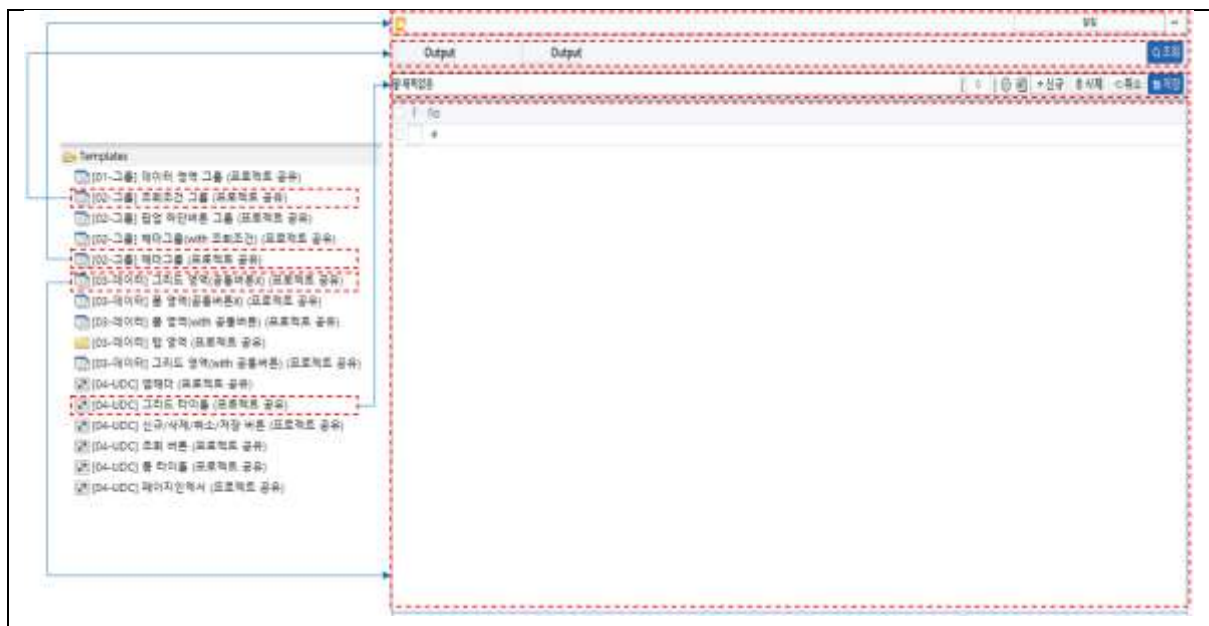
**자주 사용되는 화면 패턴을 UI 템플릿으로 등록하여 쉽게 재사용** 할 수 있습니다. 기획 단계 이후에 화면에 대한 유형별 패턴 및 컴포넌트에 대한 공통 요소 템플릿화를 진행하여 개발자에게 공유하여 사용할 수 있습니다. UI 템플릿으로 등록된 패턴 그룹은 Palette 에 등록되며 일반 컨트롤처럼 쉽게 재사용 할 수 있습니다. UI 템플릿으로 등록하는 방법은 디자인 뷰에서 템플릿으로 등록할 컨트롤 선택 후 우 클릭하고 UI 템플릿으로 등록 항목을 선택하여 프로젝트별로 UI 템플릿을 관리할 수 있습니다.

프로젝트 전용 설정 구성을 체크하면 .settings 폴더에 canned-templates.xml 파일이 생성되며 추가된 내용이 작성됩니다.





[자주 사용하는 패턴의 컴포넌트 및 그룹영역을 UI 템플릿으로 등록]



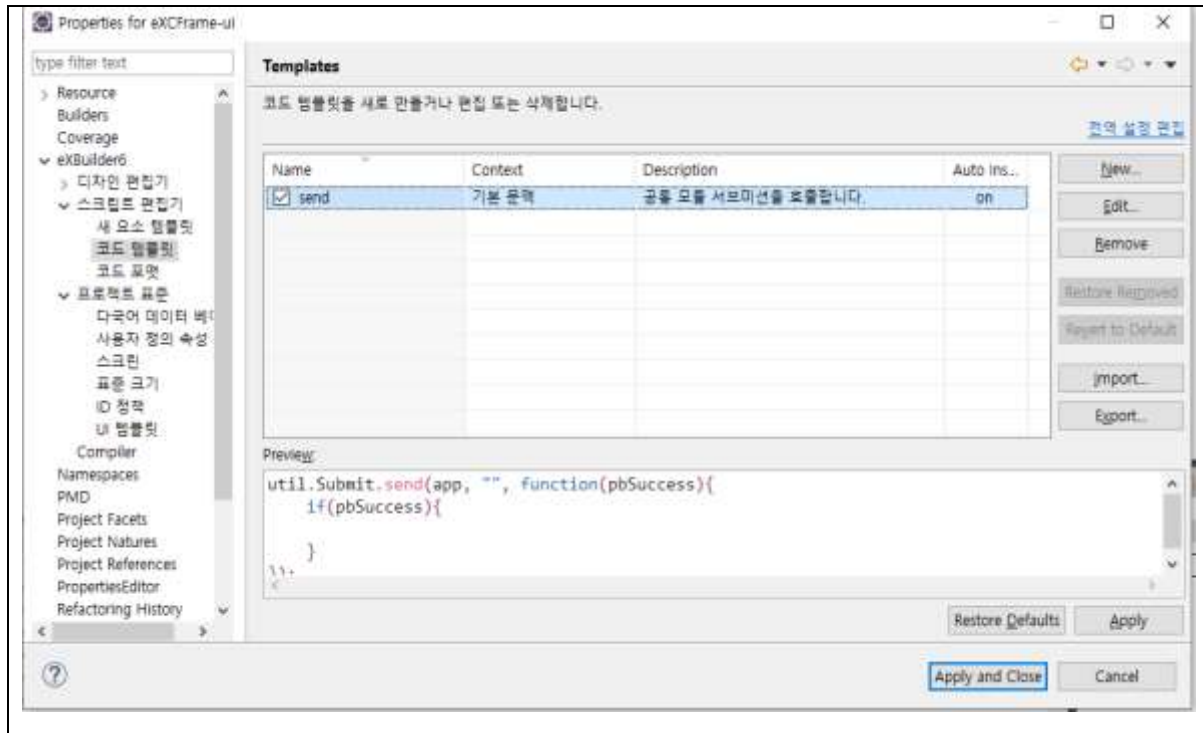
[UI 템플릿으로 등록된 패턴을 이용한 화면 디자인]

### 2.1.5. 코드 템플릿

스크립트 편집기에서 사용하고자 하는 일정한 형식을 가진 함수나 코드 형식을 템플릿으로 추가할 수 있습니다.

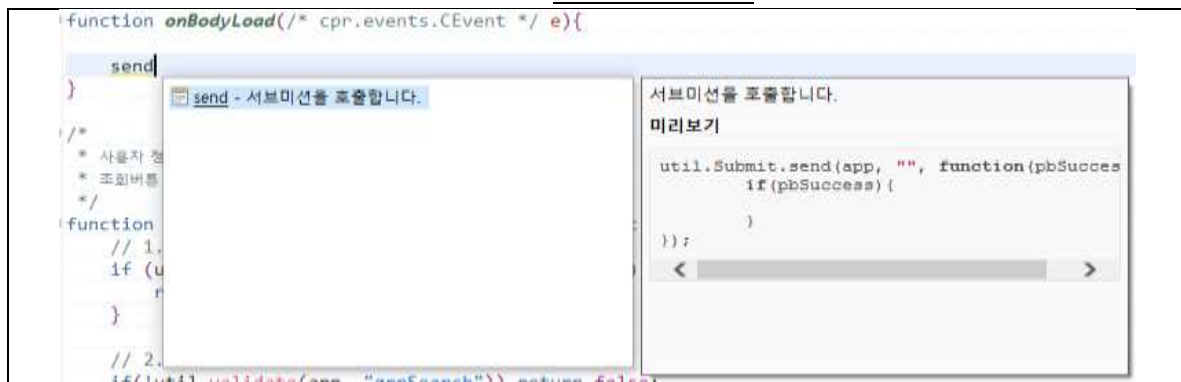
공통담당자는 프로젝트 특성에 맞춰 자주 사용하는 모듈 내 함수의 단축 설정 및 코드 어시스트 기능을 지원하기 위해 공급할 수 있습니다.

프로젝트 전용 설정 구성을 체크하면 .settings 폴더에 kr.co.tomatosystem.codetemplate.prefs 파일이 생성되며 추가된 내용이 작성됩니다.



[Properties &gt; eXBuilder6 &gt; 스크립트 편집기 &gt; 코드 템플릿]

스크립트 편집기에서 추가한 코드 템플릿을 **컨텐츠 어시스트**로 입력하여 사용할 수 있습니다.



컨텐츠 어시스트가 호출되는 시점의 코드 문맥에 따라, 템플릿들은 분류를 가질 수 있으며, 이를 **템플릿 문맥**이라고 합니다.

템플릿 문맥 유형	설명
기본 문맥	일반적인 코드 문맥으로, 어디서든 사용 가능한 문맥일 경우 이 문맥을 사용하며 다른 코드 문맥에 일치하지 않는 경우에만 템플릿을 사용하려면 이 문맥을 사용합니다.
멤버 문맥	특정 타입의 인스턴스 객체의 멤버 변수 또는 멤버 메서드를 호출할 때, 사용자가 지정한 멤버 객체 다음에 이 문맥을 사용합니다.
정적 멤버 문맥	특정 타입의 정적 멤버 변수 또는 멤버 메서드를 호출할 때, 사용자가 지정한 정적 멤버 객체 다음에 이 문맥을 사용합니다.

생성자 문맥	새로운 객체를 생성할 때, new 키워드 다음에 이 문맥을 사용합니다.
-----------	---

## 2.2. 환경 설정

### 2.2.1. env.json

프로젝트 내 필요한 정적 리소스의 레퍼런스를 관리하는 파일입니다.

외부라이브러리 혹은 외부 css를 등록할 수 있습니다. 전역적으로 스크립트를 삽입하거나 스타일시트를 추가해야 할 경우, 상대 경로를 작성하거나 cdn 경로를 등록하실 수 있습니다.

env.json
<pre>{   "runtime-lib" : [     "/thirdparty/echart/echarts.min.js",     "https://cdnjs.cloudflare.com/ajax/libs/echarts/5.1.2/echarts.min.js"   ],   "runtime-css" : [     "/theme/common/common-theme.less",     "/lib/loaders/loaders.min.css",     "https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"   ] }</pre>

주의할 점은 전역적으로 설정되기 때문에 전역적으로 필요한 라이브러리나 css 만 등록해주는 것이 좋습니다.

초기 화면 생성 시 서블릿 또는 컨트롤러에서 RuntimeLibFilter 나 RuntimeCSSFilter 를 이용하여 선택적으로 초기 라이브러리나 css 파일을 로드하도록 제어할 수 있습니다.

#### CleopatraUIController.java

```
/**
 * com.cleopatra.ui.PageGenerator를 이용하여 처리할 요청 URL 패턴을 정의
 */
@PostConstruct
private void initPageGenerator() {
    PageGenerator instance = PageGenerator.getInstance();
    instance.setURLSuffix(".clx");
    instance.setRuntimeLibFilter(new RuntimeLibFilter() {

        @Override
        public List<String> filterRuntimeLib(ServletContext context, HttpServletRequest request,
            List<String> runtimeLib, List<String> categorizedLib) {
            String requestURI = request.getRequestURI();
            if (requestURI.endsWith("moduleCategory.clx")) {
                // 만약 world 카테고리만 허용한다면
                List<String> libs = new ArrayList<String>();
                for (String lib : runtimeLib) {
                    System.out.println("runtimeLib :" + lib);
                    if (lib.indexOf("/user-modules.js") == -1) {
                        libs.add(lib);
                    }
                }
                for (String lib : categorizedLib) {
                    System.out.println("cateroty :" + lib);
                    if (lib.indexOf("/world.category.") != -1) {
                        libs.add(lib);
                    }
                }
                return libs;
            }
            return runtimeLib;
        }
    });

    instance.setRuntimeCSSFilter(new RuntimeCSSFilter() {

        @Override
        public List<String> filterRuntimeCSS(ServletContext context, HttpServletRequest request,
            List<String> runtimeCSS) {
            String requestURI = request.getRequestURI();
            if (requestURI.indexOf("cssFilter.clx") > -1) {
                runtimeCSS.add("app/scr/temp/outputColor.css");
            }
            return runtimeCSS;
        }
    });
}
```

Web Server 와 WAS 서버가 분리된 운영환경에서 env.json 파일은 WAS 서버에 배치되어야 하고 exbuilder.json 의 deploy-config-base 를 통해 파일의 경로를 지정 할 수 있습니다.

#### exbuilder.json 예시

```
"deploy-path": ["/ui"],
"deploy-config-base": ["WEB-INF/config/exbuilder/ui"],
```

### 2.2.2. defaults.js

프로젝트내 기본 값 및 공통 상수(속성, 스타일)를 정의할 수 있고, 스튜디오에서 바로 변경되는 것이 아니라 **런타임 시점에서 변경**됩니다. 프로젝트별 표준이 다르기 때문에 컨트롤별 default 속성을 적용하는 기능을 제공합니다.

웹프로젝트에서 Project Facets 을 통해서 eXBuilder6 Modules 를 체크하게 되면 해당 파일이 추가됩니다. 컨트롤의 properties 뷰에서 값이 **default(회색)**로 되어있는 값에서만 적용됩니다.

※ 도움말에 eXBuilder6 > 고급 기능> 공통화 및 재사용 > 기본 값 및 상수정의(defaults.js)를 참고하시기 바랍니다.

### 2.2.3. exbuilder.json

Java App Server 에 배치되고 서버 플러그인 중 eXBuilder6 관련 환경설정 정보를 관리하는 파일입니다.

#### ebuilder.json

```
{
  "deploy-path": [
    "/ui"
  ],
  "deploy-config-base": ["/WEB-INF/config/exbuilder/ui"],
  "cache-control": true,
  "html-lang": "ko",
  "page-loading-image": {
    "src": "
    "width": 250.0,
    "height": 150.0
  },
  "runtime-css": [
    "/resource/css/cleopatra.css"
  ],
  "runtime-lib": [
    "/resource/cleopatra.js",
    "/resource/conf/defaults.js"
  ],
  "fileupload": {
    "maxbodysize": -1.0,
    "maxfilesize": -1.0,
    "tempdirpath": "",
    "thresholdsize": 10240.0,
    "fileuploadhandler": null,
    "fileconvertor": null
  },
  "data-compression": false,
  "type-handler": {
    "dateformat": "yyyyMMddHHmmss"
  }
}
```

속성명	설명
deploy-path	컴파일된 eXBuilder6 웹 배포 경로
deploy-config-base	env.json 파일 배치 경로(web/was 가 분리된 환경 일 경우 작성 필요) [%DEPLOY-CONFIG-BASE%]/[%DEPLOY-PATH%]와 같은 경로에 env.json 파일이 위치해야합니다.

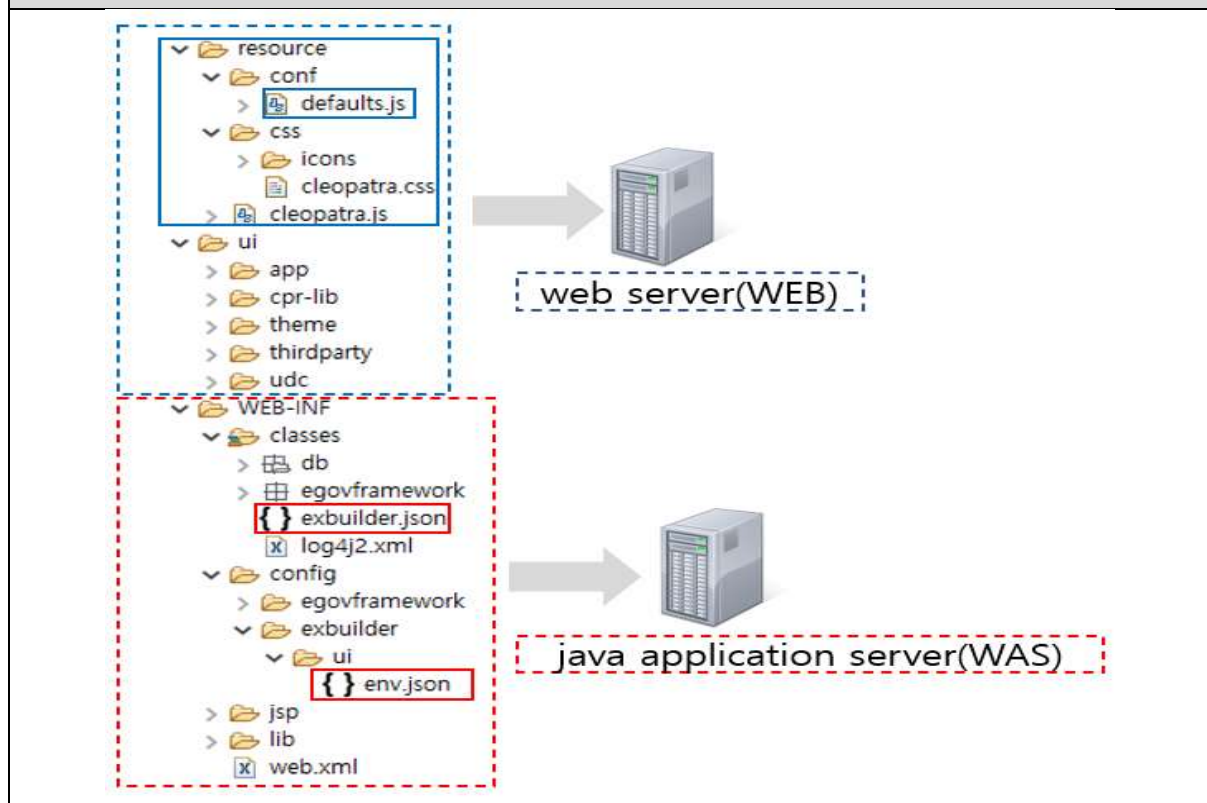
cache-control	<p>eXBuilder6 서버 플러그인에 의해 자동으로 만들어지는 HTML의 meta tag와 HTTP Response Header에 기록될 Cache Control 옵션의 사용 여부(true   false).</p> <ul style="list-style-type: none"> <li>• true : Cache 사용(default)</li> <li>• false : Cache 사용 안함</li> </ul>										
html-lang	<p>eXBuilder6 서버 플러그인에 의해 자동으로 만들어지는 HTML의 기본 lang 속성의 값(ISO 639-1 표기법). 생략시 구동된 서버의 기본 언어코드가 설정됨. ex) 한국어의 경우 : ko</p>										
runtime-lib	<p>eXBuilder6 서버 플러그인에 의해 자동으로 만들어지는 HTML의 script tag에 포함될 javascript 목록. eXBuilder6 런타임 구동에 필요한 cleopatra.js와 defaults.js는 자동으로 설정되고, 그 이외의 script 파일을 설정. 설정되는 순서로 script tag가 생성됨. script 파일의 경로는 웹 컨텍스트 경로 기준으로 설정. ex) /script/lib/jquery.min.js</p>										
runtime-css	<p>eXBuilder6 서버 플러그인에 의해 자동으로 만들어지는 HTML의 link tag에 포함될 CSS 목록. eXBuilder6 런타임 구동에 필요한 cleopatra.css는 자동으로 설정되고, 그 이외의 CSS 파일을 설정. 설정되는 순서로 link tag가 생성됨. CSS 파일의 경로는 웹 컨텍스트 경로 기준으로 설정. ex) /style/lib/jquery.ui.css</p>										
page-loading-image	<p>clx 페이지를 로딩하는 동안 출력할 로드 이미지를 설정</p> <ul style="list-style-type: none"> <li>•src : 로딩 이미지의 URL로 예시와 같이 dataUrl의 사용을 권장합니다.</li> <li>•width : 이미지의 넓이</li> <li>•height : 이미지의 높이</li> </ul> <p>위 세 속성이 모두 정확히 설정되어 있어야 로딩이미지가 정상출력됩니다.</p>										
fileupload	<p>eXBuilder6 서버 플러그인에 의해 처리되는 multipart/form-data 처리를 위한 속성 설정 그룹</p> <table border="1"> <tr> <td>maxbodysize</td><td>HTTP POST Body의 최대 용량(byte). 생략될 경우 기본설정은 -1로 무제한.</td></tr> <tr> <td>maxfilesize</td><td>업로드 되는 파일 개개의 최대 용량(byte). 생략될 경우 기본설정은 -1로 무제한.</td></tr> <tr> <td>tempdirpath</td><td>업로드된 파일을 옮길 임시 디렉토리 설정. 생략될 경우 기본 JAVA 임시 디렉토리가 설정됨.</td></tr> <tr> <td>thresholdsize</td><td>파일 업로드 처리시 최대 메모리 적재 용량을 설정(byte). 생략될 경우 기본설정은 -1로 무제한.</td></tr> <tr> <td>fileconvertor</td><td> <p>eXBuilder6 서버 플러그인은 multipart/form-data 처리를 위해 Apache Commons의 FileUpload 컴포넌트(1.3.1 이상)를 사용함. eXBuilder6 서버 플러그인은 업로드된 파일을 java.io.File 객체로 관리하는데 이외에 별도의 객체로 관리해야 할 경우</p> <p>org.apache.commons.fileupload.FileItem를 관리할 객체로 변환하는 클래스를 설정하는 데 사용하는 속성. 제공되는</p> </td></tr> </table>	maxbodysize	HTTP POST Body의 최대 용량(byte). 생략될 경우 기본설정은 -1로 무제한.	maxfilesize	업로드 되는 파일 개개의 최대 용량(byte). 생략될 경우 기본설정은 -1로 무제한.	tempdirpath	업로드된 파일을 옮길 임시 디렉토리 설정. 생략될 경우 기본 JAVA 임시 디렉토리가 설정됨.	thresholdsize	파일 업로드 처리시 최대 메모리 적재 용량을 설정(byte). 생략될 경우 기본설정은 -1로 무제한.	fileconvertor	<p>eXBuilder6 서버 플러그인은 multipart/form-data 처리를 위해 Apache Commons의 FileUpload 컴포넌트(1.3.1 이상)를 사용함. eXBuilder6 서버 플러그인은 업로드된 파일을 java.io.File 객체로 관리하는데 이외에 별도의 객체로 관리해야 할 경우</p> <p>org.apache.commons.fileupload.FileItem를 관리할 객체로 변환하는 클래스를 설정하는 데 사용하는 속성. 제공되는</p>
maxbodysize	HTTP POST Body의 최대 용량(byte). 생략될 경우 기본설정은 -1로 무제한.										
maxfilesize	업로드 되는 파일 개개의 최대 용량(byte). 생략될 경우 기본설정은 -1로 무제한.										
tempdirpath	업로드된 파일을 옮길 임시 디렉토리 설정. 생략될 경우 기본 JAVA 임시 디렉토리가 설정됨.										
thresholdsize	파일 업로드 처리시 최대 메모리 적재 용량을 설정(byte). 생략될 경우 기본설정은 -1로 무제한.										
fileconvertor	<p>eXBuilder6 서버 플러그인은 multipart/form-data 처리를 위해 Apache Commons의 FileUpload 컴포넌트(1.3.1 이상)를 사용함. eXBuilder6 서버 플러그인은 업로드된 파일을 java.io.File 객체로 관리하는데 이외에 별도의 객체로 관리해야 할 경우</p> <p>org.apache.commons.fileupload.FileItem를 관리할 객체로 변환하는 클래스를 설정하는 데 사용하는 속성. 제공되는</p>										

		<p>클래스 이외에 별도로 구현할 경우</p> <p>com.cleopatra.protocol.parser.FileConvertor 를 구현해야 함. 설정은 구현된 클래스의 패키지명을 포함한 전체 이름을 등록. 설정된 클래스에서 변환한 객체를 반환받기 위해서는 com.cleopatra.protocol.data.DataRequest 의 getFileObjects 메소드를 통해 직접 얻어내거나 getUploadFiles 메소드를 통해 얻은 com.cleopatra.protocol.data.UploadFile 객체의 getFileObj 메소드를 통해 획득 가능. 생략할 경우 com.cleopatra.protocol.parser.DefaultFileConvertor 가 동작하고 java.io.File 객체로 관리됨.</p> <p>▷ Springframework 의 org.springframework.web.multipart.MultipartFile 로 관리가 필요한 경우 "com.cleopatra.spring.SpringFileConvertor" 로 설정할 수 있음.</p>
	fileuploadhandler	<p>com.cleopatra.protocol.parser.fileupload.FileUploadHandler 인터페이스를 구현한 클래스로 eXBuilder6 서버 플러그인에 내장된 핸들러, 상세내용은 <a href="#">10.5.1 파일업로드 목차 참고</a></p>



## 2.2.4. 디플로이 환경 파일 위치

### 디플로이 시 환경설정 파일 위치 정보



## 3. 앱

### 3.1. 앱

앱이란 한쌍의 `clx, js` 파일이 컴파일 되어 구성하는 하나의 프로그램(`clx.js`)을 의미합니다.

앱은 프로그램의 목적에 따라 일련의 UI 컨트롤 및 데이터 컨트롤들을 구성하고 이들을 연결하여 제어하는 방법으로 구성될 수 있습니다.

모든 앱은 앱 URI 또는 앱 ID 라는 식별자를 가지며 **독립적인 Scope** 를 가집니다. 앱의 ID 는 소스 폴더로부터의 상대 경로가 되며 파일 이름 중에 `.clx` 확장명은 생략됩니다. 예를 들어 `clx-src/app/test.clx` 의 경우 `app/test` 가 앱의 ID 가 됩니다.

```
var myApp = new cpr.core.App("앱의 식별자", {
    onCreate: function( /* cpr.core.AppInstance */ app, exports) {
        app.getContainer().style.css({
            width: "300px",
            height: "200px",
            "background-color": "orange"
        })
        var button = new cpr.controls.Button();
        button.value = "Hello World";
        app.getContainer().addChild(button, {
            left: "10px",
```



```

        top: "10px",
        width: "100px",
        height: "25px"
    });
}
});
cpr.core.Platform.INSTANCE.register(myApp);

```

[화면에 버튼을 추가하는 애플리케이션의 예시]

마지막 줄에서 플랫폼에 앱을 등록하는 코드를 볼 수 있습니다. 이 때 앱의 식별자를 이용하여 플랫폼에 앱을 등록하게 됩니다. 스튜디오에서 위치웍으로 clx 파일을 편집하는 경우, 위와 같은 자바스크립트 코드로 컴파일 됩니다.

### 3.2. 앱 인스턴스

앱을 실행하려면, 앱을 이용하여 새로운 앱 인스턴스를 만들고 실행해야 합니다. 앞선 예시의 앱을 실행하려면 다음과 같이 합니다.

```

var appInstance = myApp.createNewInstance();
appInstance.run();

```

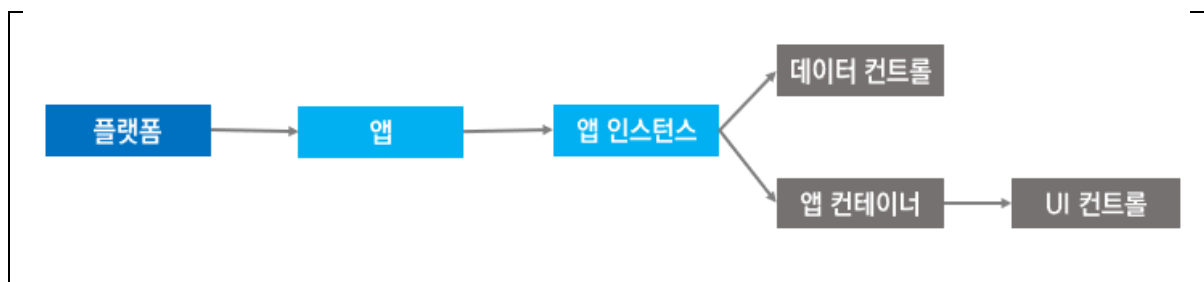
모든 앱 인스턴스는 자동으로 루트 컨테이너(`cpr.controls.Container`)를 갖게 되며, 이곳에 UI 컨트롤들이 부착됩니다. 어떠한 경로를 통해서든 이 루트 컨테이너에 포함된 모든 컨트롤은 앱 인스턴스에 귀속되며, `Control.getAppInstance()`를 통해 앱 인스턴스에 접근할 수 있습니다. 반면 데이터 컨트롤들은 UI 컨트롤이 아니므로, 별도로 아래와 같이 앱 인스턴스에 등록해 주어야 나중에 `app.lookup()`으로 찾을 수 있게 됩니다. 등록 과정은 필수적인 것은 아니지만, 나중에 ID로 참조하려면 등록 절차가 필요합니다.

```

var dataSet_1 = new cpr.data.DataSet("ds1");
dataSet_1.parseData({
    ...
});
// 데이터셋 컨트롤을 앱 인스턴스에 등록
app.register(dataSet_1);

```

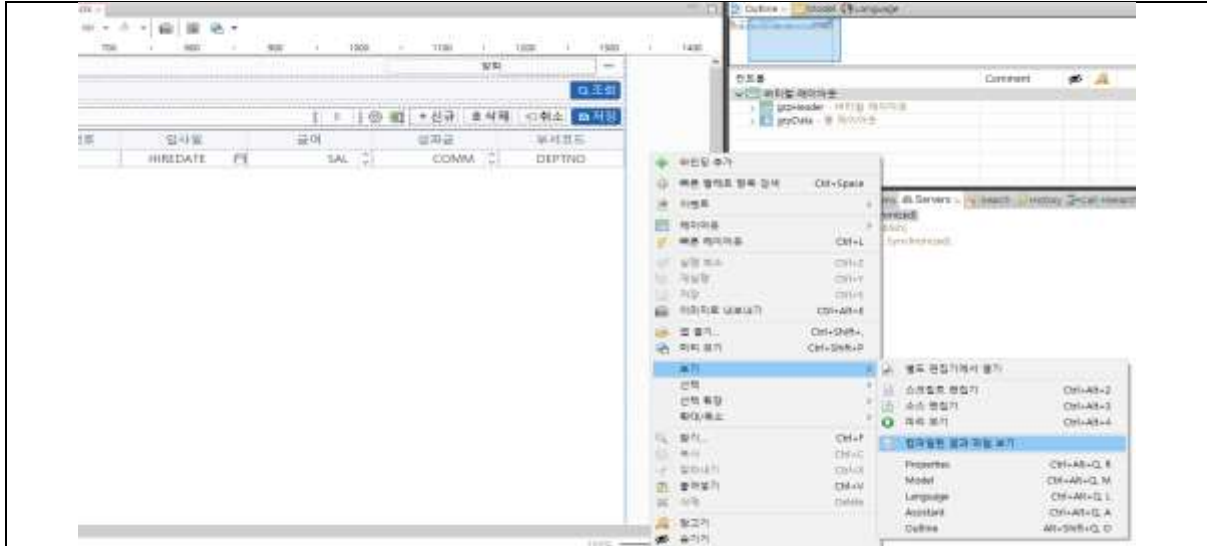
[데이터셋 컨트롤을 앱 인스턴스에 등록하는 예시]



[앱 인스턴스의 UI 컨트롤 및 데이터 컨트롤 배치 구조]

**참고)**

스튜디오에서 위지윅으로 편집된 clx 파일의 컴파일된 결과물은 디자인 탭에서 우클릭 -> 보기 -> 컴파일된 결과보기(clx.js)로 확인 가능하므로 앱, 컨테이너, UI 컨트롤등의 상관관계를 이해하시는데 도움이 될 것입니다.

**3.3. 앱 속성 출판**

앱 안에 다른 앱을 포함시켜 구현할 수 있습니다. 실행 중인 앱 인스턴스는 외부에서 접근할 수 있는 속성들을 정의할 수 있으며, 이를 **앱 속성**이라 칭합니다.

앱 속성은 Assist 뷰에 출판된 속성에서 추가할 있고, 일반적으로는 공용적으로 사용 될 **UDC 컨트롤 및 임베디드 앱 컨트롤에 배치될 앱에서 주로 사용됩니다.**

Assist 뷰에서 출판된 속성들의 목록을 확인하거나 편집할 수 있습니다. 출판된 앱 속성들은 외부에서 값을 얻거나 지정할 수 있으며, 외부에서 값을 변경하는 경우, 앱 인스턴스는 property-change 이벤트를 전파합니다.

※ UDC 앱 속성 관련하여 항상 property-change 가 일어나는 것은 아니며 특정 조건에 따라 발생하게 됩니다. 그리드내 UDC 가 배치되고 UDC 에 상대 컬럼 바인딩이나 앱 속성 바인딩을 지정하였을 경우 이벤트 발생이 되지 않을 경우가 있습니다.

유형	설명
string	문자열
color	컬러 피커를 통해 지정 가능한 HTML code(#000000)
number	숫자 형식
boolean	참/거짓 형식
control	컨트롤 형식
object	오브젝트 형식
resource	리소스의 경로 형식

<b>enumeration</b>	Enum 형식 (콤보박스 구조)
<b>expression</b>	표현식 형식

```
/* 앱의 속성을 가져옵니다.*/
app.getAppProperty("property");
/*앱의 속성에 값을 설정합니다.*/
app.setAppProperty("property", "propertyValue");
```

### 3.4. 앱 이벤트 출판

앱 속성과 마찬가지로 Assist 뷰에 출판된 이벤트에서 앱이 제공하는 이벤트를 정의할 수 있습니다. 모든 앱들은 자기 자신만의 이벤트를 전파할 수 있습니다.

```
/* 이벤트를 생성합니다.*/
var event = new cpr.events.CUIEvent("eventName");
/* 이벤트를 전파합니다.*/
app.dispatchEvent(event);
```

반드시 Assist 뷰에서 선언하지 않더라도 스크립트로 이벤트를 전파할 수 있으며 외부에서 스크립트 코드로 이벤트를 읽을 수 있습니다. 하지만 스튜디오에 디자인 뷰에서 이벤트 추가 기능을 사용할 수 없습니다. 이렇게 전파된 이벤트는 외부에서 일반 컨트롤이 가지고 있는 이벤트처럼 사용할 수 있습니다.

### 3.5. 앱 함수 출판

앞서 설명한 것처럼 모든 앱은 독립적인 **scope** 를 가지고 있습니다. 그렇기 때문에 출판되지 않은 변수나 함수들은 외부에서 접근이 불가능하며 각각의 앱의 변수 및 함수들은 서로 충돌을 일으키지 않습니다. 외부에서 호출할 수 있는 함수를 제공하려면 **exports** 키워드를 통해 함수를 출판해야 합니다. 이렇게 출판된 함수는 **callAppMethod()** 함수를 통해서 외부에서 호출하여 사용할 수 있습니다.

#### 출판 예시

```
/**
 * 알람메시지
 * @param {String} message 메시지
 */
exports.alertMsg = function alertMessage(message){
    alert(message);
}
```

#### 호출 예시

```
호출 할 앱인스턴스.callAppMethod("alertMessage", ["메시지"]);
```

## 4. 임베디드 앱

임베디드 앱은 다른 앱을 추가 할 수 있는 컨트롤입니다.

임베디드 앱은 프로젝트 내 빈번히 사용하여 재활용 가능한 앱 또는 MDI 폴더, 탭폴더 내 앱으로 많이 사용됩니다.

아래의 예시는 특정한 앱을 로드한 뒤 임베디드 앱의 인스턴스를 교체하는 과정입니다.

```
/** @type cpr.controls.EmbeddedApp */
var emb = app.lookup(psEmbeddedappId);
if(emb){
    cpr.core.App.load(psAppId, function(loadedApp){
        if(loadedApp){
            /*로드된 앱을 임베디드 앱에 설정*/
            emb.app = loadedApp;
            emb.ready(function(e){
                /*통신 전달값*/
                emb.initValue = polnitValue;
            })
        }
    });
}
```

cpr.core.App.load 함수는 플랫폼에 등록되어 있는 앱의 경우 즉시 콜백을 호출하며 그렇지 않은 경우에는 앱을 로드한 후에 콜백을 호출합니다.

임베디드 앱은 실제로 화면에 그려져 표시해야 할 필요가 있을 경우에만, 주어진 앱의 인스턴스를 자동으로 만들고 실행합니다. 이 때, **app-ready** 이벤트를 전파합니다. 임베디드 앱이 만들어진 후, 한 번도 화면에 표시되지 않는다면 앱 인스턴스는 실행되지 않습니다. 이는 높은 성능을 위함이며 앱의 인스턴스를 만들고 실행하는 동작을 꼭 필요할 때까지 자동으로 미룹니다.

### 4.1. 임베디드 앱 인스턴스 접근

일반적인 경우에는 app 을 통해서 인스턴스에 접근할 수 있었지만 임베디드 앱 내의 앱 인스턴스를 접근할 때는 **embeddedApp. getEmbeddedAppInstance()** API 를 이용하여 접근할 수 있습니다. 만약 앱 인스턴스가 만들어져 실행된 상태가 아니라면 null 을 리턴합니다.

임베디드 앱의 **ready()** 함수는 임베디드 앱이 만들어진 후 실행되기 직전에 호출됩니다. 이 때, 앱이 실행되기 전에 필요한 초기화 작업(앱 속성, 초기값 세팅)등을 수행할 수 있습니다. 또한 앱 속성을 통해 파라미터 형식의 인자값을 전달해 줄 수 있습니다.

```
/** @type cpr.controls.EmbeddedApp */
var emb = app.lookup(psEmbeddedappId);
emb.ready(function(ea) {
    ea.setAppProperties({"properties" : "propertiesValue"});
    ea.callAppMethod("method", "args");
});
```

## 4.2. 임베디드 앱에서 부모 앱 인스턴스 접근

앱 인스턴스의 `getHost()` 메서드를 이용하면, 자기 자신의 앱 인스턴스를 사용하는 임베디드 앱 컨트롤을 얻을 수 있습니다. 임베디드 앱은 항상 외부 앱 인스턴스에 포함되어 있으므로 `getHost().getAppInstance()`를 이용하여 부모 앱 인스턴스에 접근할 수 있습니다.

```
var hostAppIns = app.getHostAppInstance();
hostAppIns.setAppProperties({"properties" : "propertiesValue"});
hostAppIns.callAppMethod("method", "args");
```

## 4.3. 최상위 앱 인스턴스 접근

임베디드 앱의 부모는 다른 임베디드 앱일 수도 있습니다. SPA 구조일 경우 최상위 앱 인스턴스를 얻기 위해 `getRootAppInstance()` API 사용해 최상위 루트 앱 인스턴스에 접근할 수 있습니다.

```
app.getRootAppInstance();
```

## 4.4. 임베디드 앱 강제 실행

임베디드 앱은 실제로 화면에 그려져야 할 때 화면을 load 합니다. 탭 폴더 및 MDI 폴더의 경우 활성화 되지 않은 탭은 앱이 실행되지 않은 상태입니다. 만약 앱내 특정 데이터 및 컨트롤을 취득해야 할 경우 `forceRun()` API 또는 `forceRun` 속성을 통해 아직 load 되지 않은 앱을 강제로 실행시킬 수 있습니다.

속성	값	
임베디드 앱		
comment		
id	ea2	
removeOnBuild	false	
fieldLabel		
tooltip		
visible	true	
enabled	true	
readOnly	false	
src	ea2.clx	
forceRun	false	<pre>var embeddApp = app.lookup("임베디드ID"); embeddApp.forceRun();</pre>

## 5. UDC

UDC는 사용자 정의 컨트롤(User Defined Control)로, 공통 담당자들이 프로젝트의 특수한 상황에 맞추어, 프로젝트에서 요구되는 **공용 컨트롤을 직접 작성하여 공급**할 수 있습니다. UDC의 작성과 소비 전 과정에서 스튜디오의 지원을 받을 수 있습니다. UDC를 생성하는 방법은 반드시 소스 경로 아래 **udc 폴더** 밑에 새로운 화면을 생성하여 작성하면 됩니다.

UDC는 eXBuilder6가 자체적으로 제공하는 컨트롤과 완전히 동일한 조건으로 제공됩니다.

※ 도움말에 eXBuilder6 > 고급 기능 > 공통화 및 재사용 > 사용자 정의 컨트롤(UDC)을 참고하시기 바랍니다.

### 5.1. 그리드 내에서 사용자 정의 컨트롤(UDC) 사용 방법

그리드 내부에 UDC 가 존재할 경우, 성능상의 문제로 인해 그리드 편집 모드로 변경되지 않은 상태에서는 UDC 의 디자인을 확인할 수 없습니다. 따라서 그리드 뷰잉 모드일 경우, 단순히 컬럼 값의 텍스트 형태로 표기됩니다. 아래는 그리드 뷰잉 모드 시, 표시할 텍스트를 지정하는 예시입니다. 참고로 UDC 파일 생성 시, getText() 메서드는 자동 생성됩니다.

```
/**
 * UDC 컨트롤이 그리드의 뷰 모드에서 표시할 텍스트를 반환합니다.
 */
exports.getText = function(){
    return app.lookup("ipbPartyId").value;
};
```

### 5.2. 개발 유의점

- UDC 로 개발되는 컴포넌트는 단순한 비즈니스 로직과 UI 로 구성되어야 합니다.
- 여러 화면에서 사용하기 위해(임베디드 형태로 구현한) 개발된 화면은 UDC 로 구현하지 않고, 임베디드 앱으로 개발합니다. (그리드 및 입력 폼 형식의 앱은 지양)
- UDC 는 앱 로드 시 하나의 컨트롤처럼 로딩 되고 UDC 내 init 이나 load 이벤트에서 비동기 통신을 통하여 취득하는 데이터로 출판된 속성의 값을 지정하거나 호스트 앱과의 통신 매개체로 사용할 때에는 유의해서 사용해야 합니다. UDC 가 비동기일 경우, 비동기가 수행되고 난 이후에 호스트 앱과 통신해야 합니다.

### 5.3. 임베디드 앱과 UDC 의 차이점

	사용자 정의 컨트롤(UDC)	임베디드 앱
로딩 시점	미리 준비된 앱을 하나의 컨트롤처럼 사용합니다.	iframe 처럼 앱을 화면이 로딩되는 시점에 가져와서 로딩합니다.
속성과 이벤트	<ul style="list-style-type: none"> <li>출판된 함수나 속성은 실제 컨트롤이 제공하는 함수나 변수인 것처럼 작동하게 됩니다.</li> <li>스튜디오는 위지윅 단계에서 UDC 들의 속성이나 이벤트, 그리고 함수들에 대해 콘텐츠 어시스트를 포함한 지원을 제공합니다.</li> <li>앱 속성이 바인딩 가능하도록 출판된 경우, 일반 컨트롤의 바인딩과 마찬가지로 속성명을 통한 바인딩을 수행할 수 있습니다.</li> </ul>	<ul style="list-style-type: none"> <li>동적으로 로드되기 때문에 디자인 시점에 속성이나 이벤트에 접근하여 설정할 수 없고, app-ready 이벤트를 통해 해당 앱이 모두 로드된 후 설정할 수 있습니다.</li> </ul>

사용 예	프로젝트 내에 전역적으로 많이 사용하는 컨트롤 규칙 등을 UDC 로 만들어 재사용하며 좀 더 편리하고 빠르게 개발이 가능합니다.	화면에 보여지는 앱을 동적으로 교체하는 등의 페이지를 구성 시 임베디드 앱을 사용합니다.
------	---	---

## 6. 외부 연동

eXBuilder6 는 외부 페이지 혹은 서드파티 제품을 사용할 수 있게 해주는 컨트롤을 제공합니다.

### 6.1. 임베디드 페이지

외부의 웹페이지(html, jsp, pdf 등)를 앱에 포함하여 보여주는 컨트롤입니다. Iframe 구조로 페이지를 불러오며, 불러온 페이지에서 제공 함수를 호출 할 수 있습니다.

#### 6.1.1. 임베디드된 페이지의 함수 호출 및 파라미터 전송하는 방법

##### jsp 파일 예제

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
    function getInputText(){
        if(_ownerApp){
            setJspText();
        } else {
            alert("app getOutputText.");
        }
    }

    function setAppText(){
        if(_ownerApp) {
            _ownerApp.lookup("optGetVal").value = document.getElementById("jspInput").value;
        } else {
            alert("app setAppText.");
        }
    }

    function setJspText(){
        document.getElementById("jspInput").value = _ownerApp.lookup("ipbSetVal").value;
    }
</script>
</head>
<body>
<h2>JSP 화면</h2>
<input id="jspInput" type="text" value="Test"/>
<button onClick="setAppText()">clx로 데이터 보내기</button>
<button onClick="getInputText()">clx에서 데이터 가져오기</button>
</body>
</html>
```

다음과 같은 스크립트를 작성하게 되면 임베디드 된 페이지(jsp) 화면에서 app 을 가지고 와서 앱 내부의 컨트롤의 속성을 호출하거나 변경할 수 있습니다.

```
function onEp1Load(/* cpr.events.CEvent */ e){
    /**
     * @type cpr.controls.EmbeddedPage
     */
    var ep1 = e.control;
    ep1.setPageProperty("_ownerApp", app);
}
```

다음과 같은 스크립트를 작성하면 임베디드된 페이지(jsp) 화면에 있는 함수를 불러올 수 있습니다. callPageMethod 의 경우 도메인이 같을 경우에만 사용할 수 있습니다.

```
if(ea.hasPageMethod("alertMessage")) {
    ea.callPageMethod("alertMessage", "안녕하세요");
}
```

도메인이 다른 경우, 윈도우 객체에 postMessage API 와 이벤트 리스너를 추가하여 JSON 이나 string 값을 넘겨줄 수 있습니다. 자세한 내용은 MDN 링크를 참고하시기 바랍니다.

<https://developer.mozilla.org/ko/docs/Web/API/Window/postMessage>

## 6.2. 웹 컨트롤

eXBuilder6 는 외부 라이브러리를 이용할 수 있는 컴포넌트인 웹 컨트롤을 제공합니다.

eXBuilder6 에서는 DOM 에 대한 접근을 제한하고 있지만 웹은 DOM 을 제공하여 외부 라이브러리와 연결이 가능합니다.

차트, 에디터 등의 서드파티 라이브러리를 웹 컨트롤에 등록하여 컴포넌트로 활용할 수 있습니다.

이러한 라이브러리를 사용할 때에는 **env.json** 에 **외부라이브러리를 등록**하거나 스튜디오의 **Assist** 뷰에 **외부 스크립트를 통해서 등록**할 수 있습니다. 웹 컨트롤은 **init** 과 **load 이벤트**를 통해서 라이브러리를 설정합니다.

아래는 eChart 에서 bar 차트를 그리는 예제입니다.

### 웹 Init() 이벤트

```
/*
 * 웹에서 init 이벤트 발생 시 호출.
 */
function onShl1Init(/* cpr.events.CUIEvent */ e){
    /**
     * @type cpr.controls.UIControlShell
     */
    var shl1 = e.control;
```



```

/*최초 init 이벤트에서 e.content는 null입니다. 2번째 이벤트부터 DOM을 갖고 있습니다.*/
if(e.content) {
    /*다음에 일어나는 쉘 load이벤트 발생을 방지합니다.*/
    e.preventDefault();
}

window.addEventListener("resize", function(e){
    if(!app.disposed){
        app.lookup("shl1").getComponent("bar").resize();
    }
});
}

```

### 쉘 load() 이벤트

```

/*
 * 쉘에서 load 이벤트 발생 시 호출.
 */
function onShl1Load/* cpr.events.CUIEvent */ e){
    /**
     * @type cpr.controls.UIControlShell
     */
    var shl1 = e.control;
    var voContent = e.content;
    /*e.content가 없으면 그리지 않습니다.*/
    if(!voContent) {
        return;
    }
    /* 쉘에 'eChartContent'라는 이름으로 컴포넌트를 등록합니다. */
    shl1.registerComponent("eChartContent", voContent);
    /* 차트를 그립니다.*/
    drawChart();
}

/**
 * bar 차트를 그립니다.
 * @param {any} poContent
 */
function drawChart () {

    var poContent = app.lookup("shl1").getComponent("eChartContent");
    voChart = echarts.init(poContent);
    /** @type cpr.data.DataSet */

```

```

var vcDataset = app.lookup("dataSet");
if(vcDataset){

    var voOption = {
        xAxis: {
            type: 'category',
            data: vcDataset.getColumnData("COLUMN1")
        },
        yAxis: {
            type: 'value',
        },
        series: [{
            data: vcDataset.getColumnData("COLUMN2"),
            type: 'bar'
        }]
    };
    voChart.setOption(voOption);
}
}

```

만약에 셀 컨트롤을 사용해서 여러 화면에 차트를 보여줘야하는 경우에는 UDC 를 사용하여 UDC 파일을 만들고 그 안에 셀 컨트롤을 넣어 차트를 일반적인 컨트롤처럼 사용할 수도 있습니다.

### 6.2.1. 외부 라이브러리 연동 방법

외부 라이브러리를 추가하는 방법에서는 런타임에서 불러오기, 페이지에서 불러오기, 사용자 정의 불러오기 방법이 있습니다.

#### 6.2.1.1. 런타임에서 불러오기

소스경로 내 env.json 의 내용 중 "runtime-lib"에 외부라이브러리를 작성합니다. 해당 라이브러리는 최초 런타임시 함께 로드됩니다.

```

{
    "runtime-lib": [
        "/Lib/codemirror-5.60.0/mode/javascript/javascript.js",
        "/Lib/html2canvas/html2canvas.min.js"
    ],
    "runtime-css": [
        "/theme/cleopatra-theme.less",
        "/theme/custom-theme.less",
        "/Lib/codemirror-5.60.0/Lib/codemirror.css",
    ]
}

```

### 6.2.1.2. clx 페이지에서 불러오기

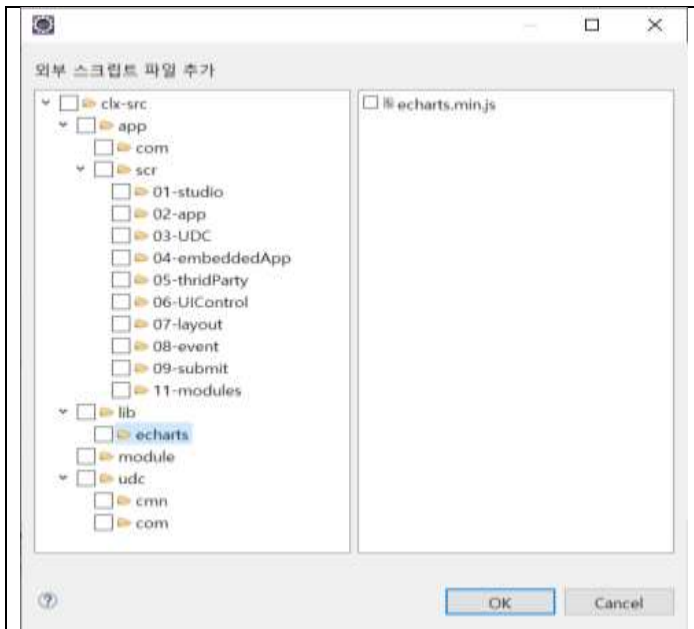
clx 의 디자인 편집에서 Assist 뷰에 외부 스크립트 탭에서 외부 라이브러리를 추가할 수 있습니다. 페이지에서 불러오는 경우 내부 리소스 및 프래그먼트/호스트로 엮인 리소스만 추가할 수 있습니다.

먼저 프로젝트에 필요한 외부 라이브러리가 소스 경로 내에 위치해야 합니다.

디자인 편집기에서 Assist 뷰에 외부 스크립트 탭에서 외부 스크립트 파일 추가 버튼을 클릭합니다.



추가할 라이브러리를 선택합니다.



### 6.2.1.3. 사용자 정의 불러오기

다음은 사용자 정의 외부 라이브러리 추가하는 방법입니다. 리소스 로더를 사용하여 특정 clx 를 로드할 때 외부 라이브러리를 추가할 수 있습니다.

#### ■ 리소스 로더

리소스 로더는 특정 앱, 외부 스크립트, CSS 자원 및 JSONP 프로토콜을 통한 데이터 등을 로드하고 준비시킨 후 코드를 실행할 수 있도록 도와줍니다.

```
/*
 * 루트 컨테이너에서 init 이벤트 발생 시 호출.
 * 앱이 최초 구성될 때 발생하는 이벤트 입니다.
 */
function onBodyInit(/* cpr.events.CEvent */ e){
    if(loaded) {
```

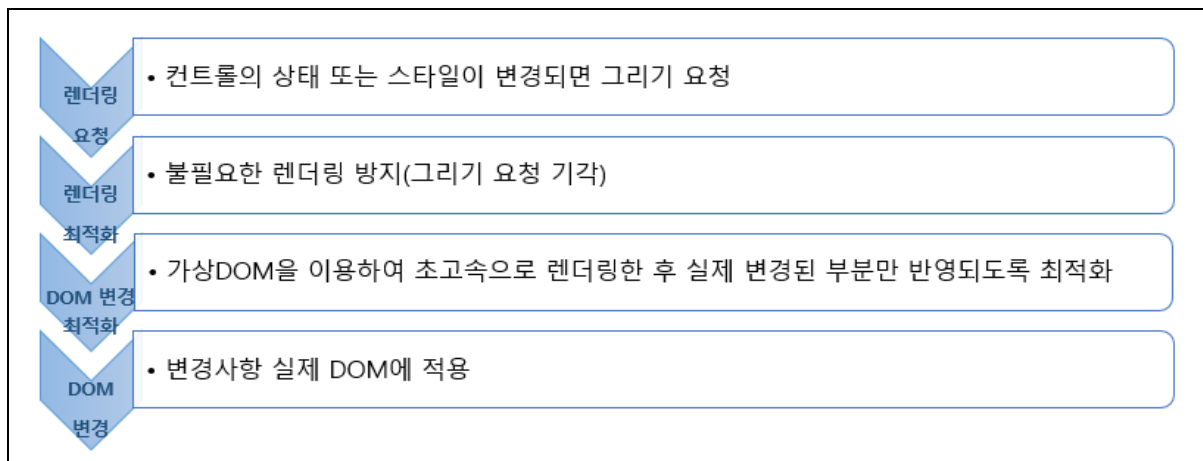
```

        return;
    }
    cpr.core.ResourceLoader.LoadScript("thirdparty/ckeditor4/ckeditor.js").then(function(input) {
        loaded = true;
        app.lookup("shlCkEditor").redraw();
    });
}

```

## 7. UI 컨트롤

### 7.1. 렌더링(그리기) 과정



컨트롤의 상태 또는 스타일이 변경되면 그리기를 요청하고, 렌더링이 예약된 컨트롤 중 불필요하게 그리기가 예약된 컨트롤의 요청을 기각합니다. 컨트롤이 제거되거나 컨트롤이 앱 인스턴스에 소속되지 않거나 컨트롤의 순서를 변경하였는데 기존의 순서와 같아 변경사항이 없을 경우, 그리기 요청이 기각됩니다. 가상 DOM 을 이용하여 초고속으로 렌더링한 후 실제 변경된 부분만 반영되도록 최적화를 거쳐 변경된 부분만 실제 DOM 에 적용합니다.

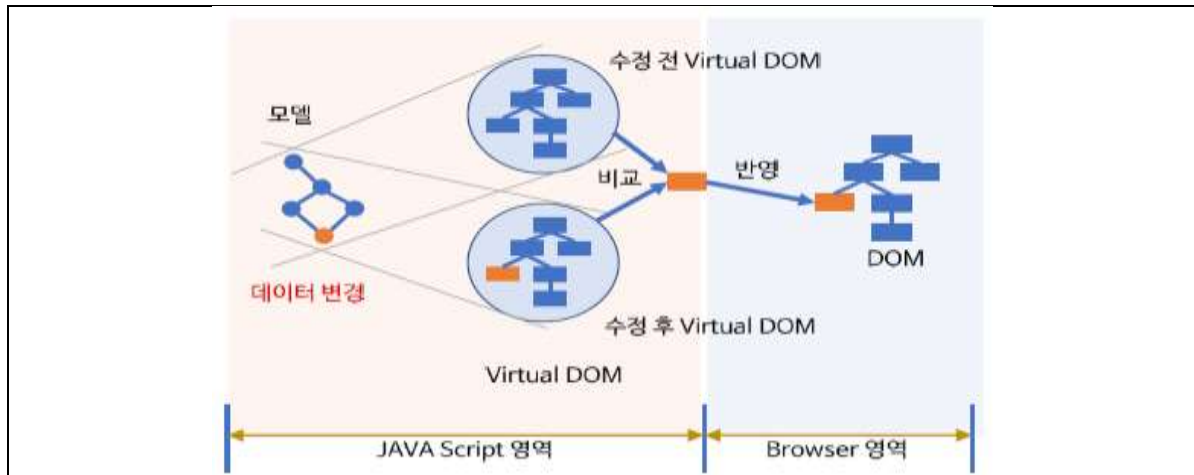
**컨트롤을 그리는 메서드로는 redraw() API 가 있습니다. 컨트롤이 다시 그려져야 될 필요가 있을 때, 다시 그리기를 요청하는 메서드이며, 실제로 새로 그리는 함수는 아닙니다.** redraw()를 호출했다고 해서, DOM 이 변경되었을 것이라고 전제해서는 안됩니다. UI 컨트롤은 어떠한 경우에도 컨트롤이 아닌, 이를 표현하는 DOM 에 직접 접근하려 시도해서는 안됩니다.

컨트롤과 연결되어 간접적으로 표시 중인 데이터의 상태가 변경될 경우 연결된 UI 컨트롤은 redraw() API 를 사용하여 다시 그려주어야 합니다. 다시 말해, 데이터 셋, 데이터 맵, 익스프레션 바인딩 등 바인딩 된 컨트롤들은 데이터를 가져온 후에 redraw() 하지 않으면 컨트롤에 보여지는 값은 갱신되지 않습니다. 하지만 예외적으로 그리드, 콤보박스 같은 경우는 redraw() 하지 않더라도 데이터가 갱신됩니다. 그리드는 연결된 데이터셋 build 시 자동으로 redraw() 하며, 콤보박스는 사용자가 콤보박스를 클릭하여 리스트가 열릴 때 redraw()가 되는 구조로 되어 있어 데이터가 갱신되더라도 redraw() API 를 호출하지 않아도 됩니다. redraw() API 를 무한히 사용하여도 실제로는 한번만 그려지게 되며 성능에 문제가 되지 않습니다.

**※ 도움말에 eXBuilder6 > 앱 개발> 렌더링(그리기) 과정을 참고하시기 바랍니다.**

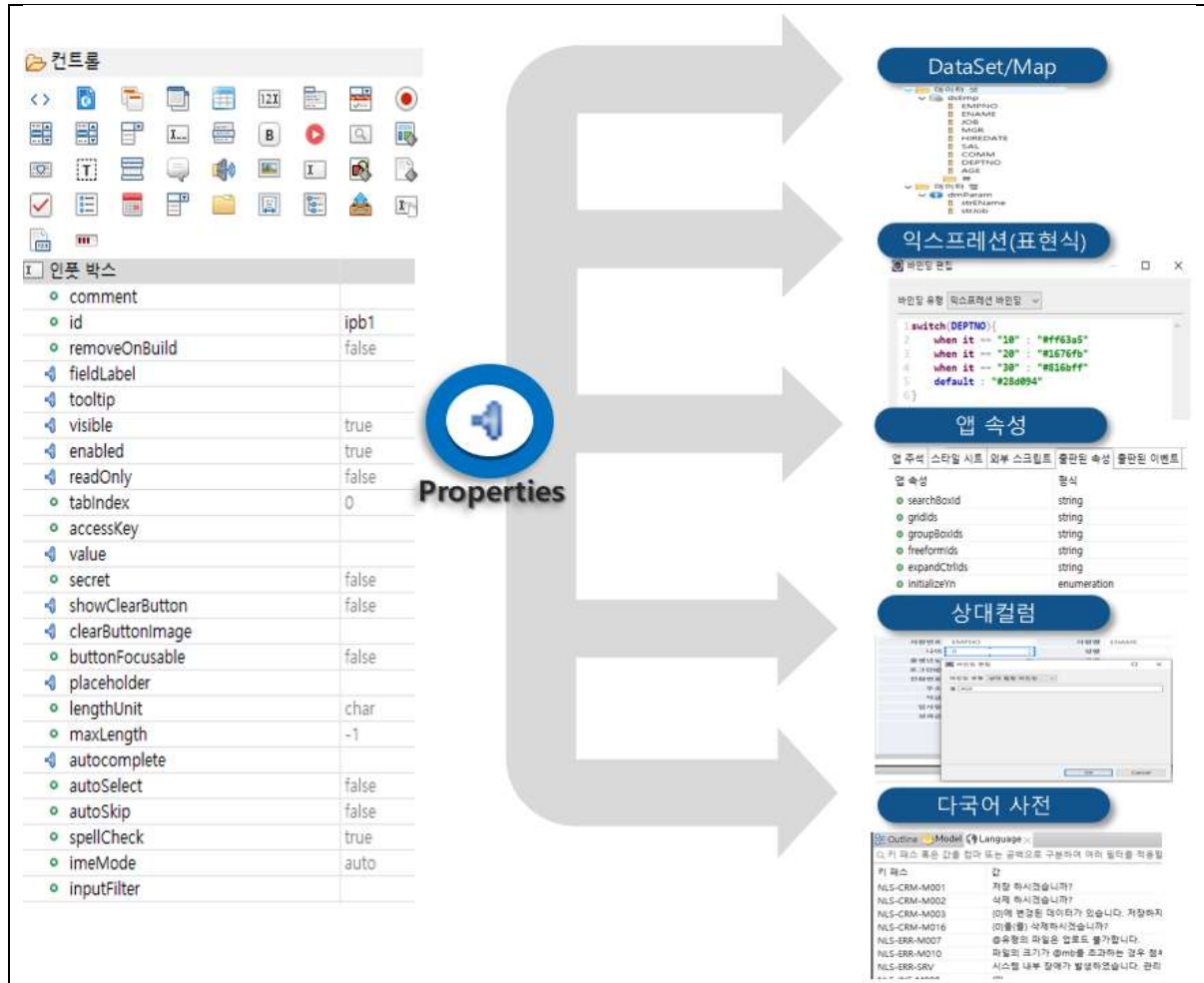
## 7.2. Virtual DOM

eXBuilder6 는 다시 그리기가 예약된 컨트롤의 범주를 최적화한 뒤, 다음 장면을 그리기 직전에 일괄처리합니다. 따라서, 컨트롤의 상태를 조작하거나 생성하거나 삭제하는 등과 같은 작업을 수행하더라도, 그리기는 한 번만 일어납니다. eXBuilder6 는 컨트롤들의 상태를 바탕으로 가상 DOM 을 먼저 구성하고, 이전 가상 DOM 과 비교하여 차이점만 실제 DOM 에 반영하게 됩니다. 새로 그려야 할 가상 DOM 을 이용하여 초고속으로 렌더링한 후 이전 DOM 과 비교해서 실제로 변경된 부분만 반영되도록 최적화합니다. 변경된 DOM 만 렌더링을 함으로써 클라이언트에서의 강력한 처리 성능을 확인할 수 있으며, 대용량 데이터 처리시에 효율성이 극대화됩니다.



### 7.3. 바인딩

바인딩을 이용하여 컨트롤의 속성, 스타일, 클래스의 값을 바인딩 소스로부터 제공받을 수 있습니다. 바인딩 소스는 **데이터 셋, 데이터 맵, 표현식, 앱 속성, 언어(다국어)**를 사용합니다. 바인딩 된 타겟 항목은 원본 값을 적용하지 않고 바인딩 소스의 값을 적용합니다.

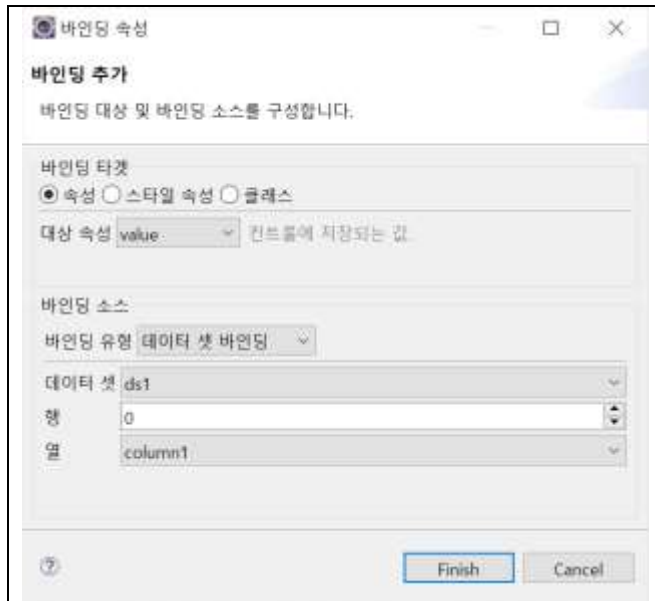


#### 7.3.1. 바인딩 유형

바인딩 소스는 데이터 셋, 데이터 맵, 표현식, 앱 속성, 언어를 사용합니다. 바인딩 소스는 연결된 바인딩 타겟에 값을 제공합니다. 바인딩 소스는 **바인딩 유형** 콤보 박스에서 선택하여 지정할 수 있습니다.

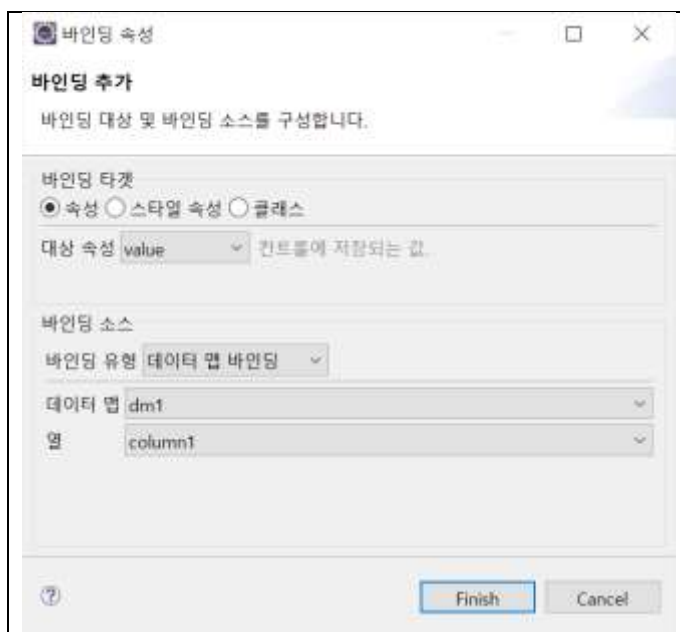
##### 7.3.1.1. 데이터 셋 바인딩

데이터 셋은 행과 열을 이용하여 바인딩 타겟에 값을 제공합니다. 모델 뷰에서 데이터 셋의 컬럼을 바인딩 타겟 컨트롤에 드래그 앤 드롭 하거나 **바인딩 편집** 대화상자에서 바인딩 유형을 데이터 셋 바인딩으로 지정한 후 데이터 셋, 행, 열을 지정하여 값을 제공합니다. **데이터 셋 목록에는 데이터 뷰도 함께 표시됩니다.**



### 7.3.1.2. 데이터 맵 바인딩

데이터 맵은 열을 이용하여 바인딩 타겟에 값을 제공합니다. 모델 뷰에서 데이터 맵의 컬럼을 바인딩 타겟 컨트롤에 드래그 앤 드롭하거나 **바인딩 편집** 대화상자에서 바인딩 유형을 데이터 맵 바인딩으로 지정한 후 데이터 맵, 열을 지정하여 값을 제공합니다.



### 7.3.1.3. 상대 컬럼 바인딩

상대 컬럼을 이용하여 바인딩 타겟에 값을 제공합니다.

**바인딩 편집** 대화상자에서 바인딩 유형을 상대 컬럼 바인딩으로 지정한 후 열을 지정하여 값을 제공합니다. 상대 컬럼은 컨트롤이나 그리드 디테일 셀에 지정할 수 있습니다.

상대 컬럼은 바인딩 타겟이 그룹이나 그리드에 포함되어 있을 경우에만 유효합니다.

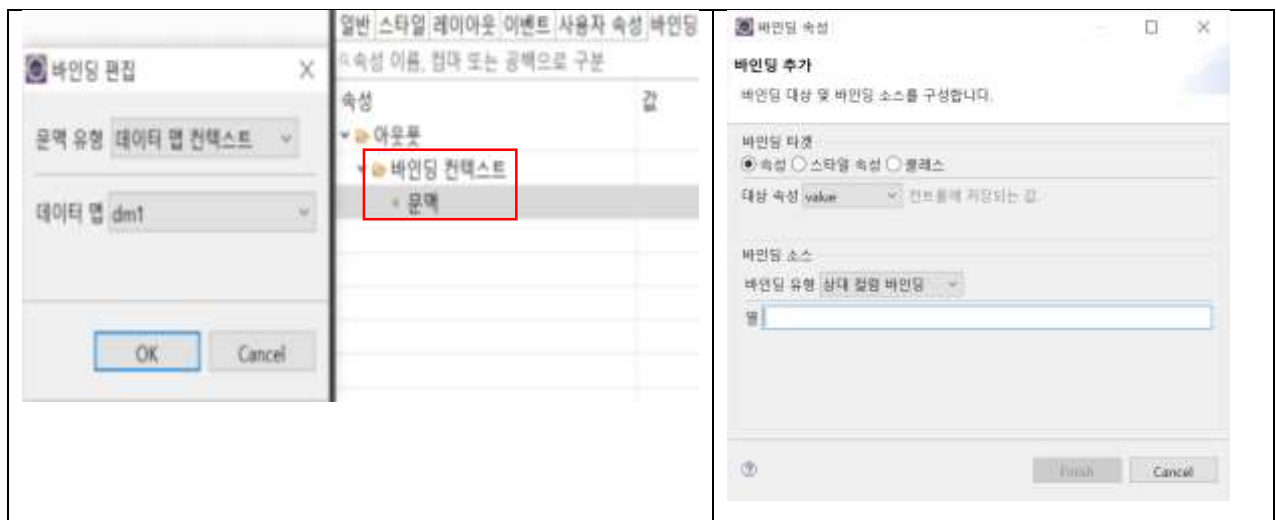
또한, 컨테이너인 그리드에 데이터 셋이나 데이터 뷰가 지정되어 있거나 그룹에 데이터 셋이나

데이터 뷰, 데이터 맵의 컨텍스트가 바인딩 되어 있어야 해당 데이터로부터 컬럼의 값을 제공합니다.

### ■ 그룹 컨트롤에 데이터 컨텍스트 지정

컨텍스트란 어떤 객체를 핸들링하기 위한 접근 수단입니다.

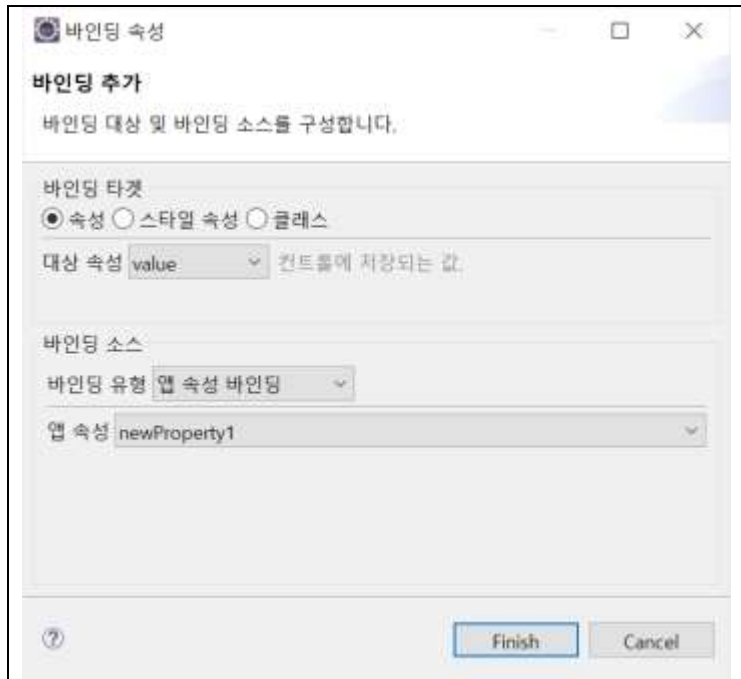
그룹 컨트롤에 데이터 컨텍스트를 지정할 수 있습니다. 데이터 컨텍스트 지정은 모델 뷰의 데이터 셋이나 데이터 뷰, 데이터 맵을 그룹에 드래그 앤 드롭하여 지정 할 수 있습니다. 또는 그룹 컨트롤을 선택한 후 [Properties - 바인딩]에서 **바인딩 컨텍스트 > 문맥** 속성의 값 셀을 더블클릭하여 열리는 **바인딩 편집** 대화상자를 이용하여 지정할 수 있습니다.



#### 7.3.1.4. 앱속성 바인딩

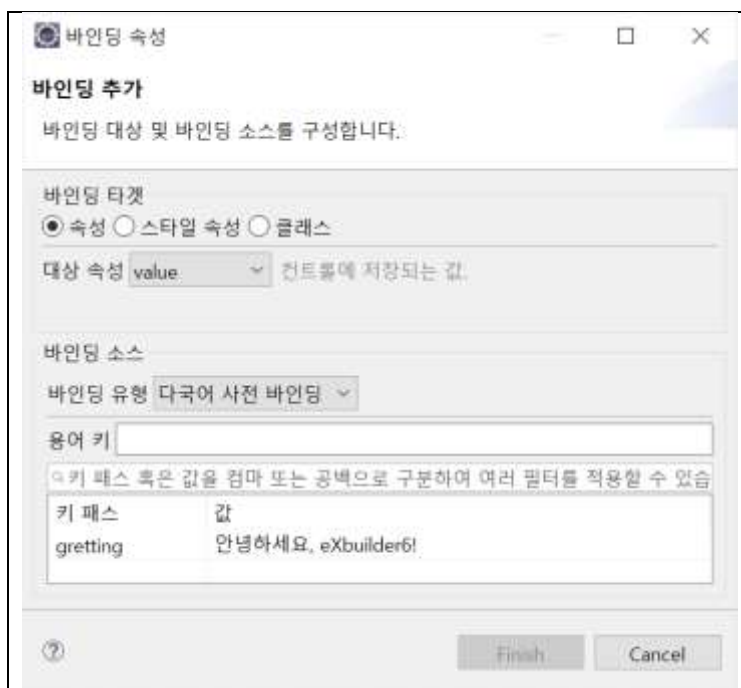
출판된 앱의 속성을 이용하여 바인딩 타겟에 값을 제공합니다. **바인딩 편집** 대화상자에서 바인딩 유형을 앱 속성 바인딩으로 지정한 후 앱 속성에 출판된 앱 속성을 지정하여 값을 제공합니다.





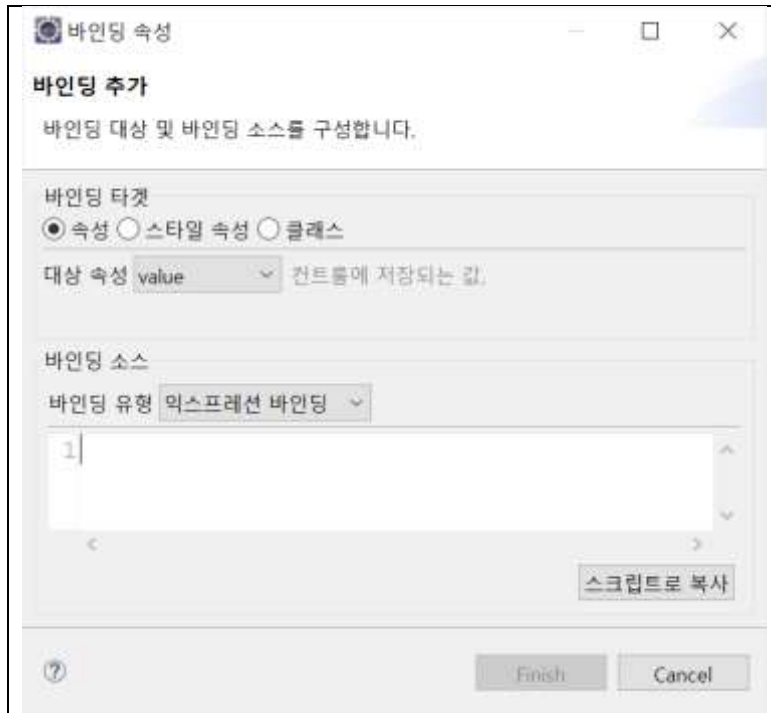
#### 7.3.1.5. 다국어 사전 바인딩

언어 파일(소스 경로/language.json)을 이용하여 바인딩 타겟에 값을 제공합니다. 언어 뷰에서 등록된 항목을 바인딩 타겟 컨트롤에 드래그 앤 드롭하거나 **바인딩 편집** 대화상자에서 바인딩 유형을 다국어 사전 바인딩으로 지정한 후 용어 키에 키 패스를 지정하여 값을 제공합니다.



### 7.3.1.6. 익스프레션 바인딩

표현식 문법을 이용하여 바인딩 타겟에 값을 제공합니다. 표현식은 상수, 함수, 연산자 등을 이용하여 값을 표현할 수 있습니다. **바인딩 편집** 대화상자에서 바인딩 유형을 익스프레션 바인딩으로 지정한 후 텍스트 박스에 표현식을 입력하여 값을 제공합니다.



### 7.3.2. 표현식 엔진

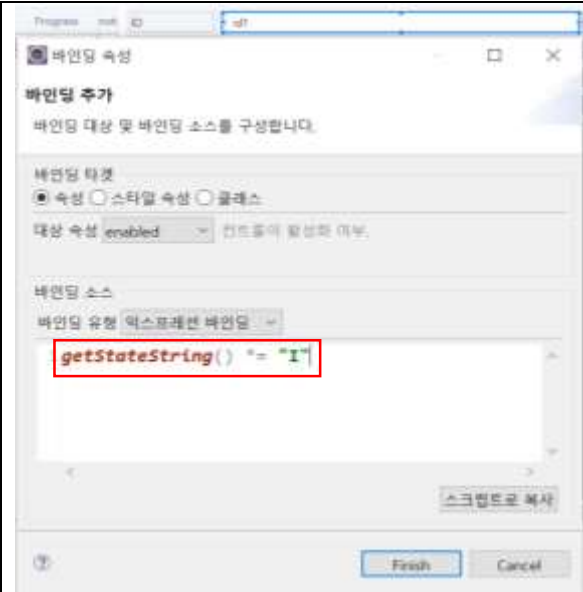
표현식 엔진(Expression Engine)은 엑셀의 수식과 같이 단일한 값을 반환하는 일종의 표현식을 수행하는 엔진이며, 다음의 영역에서 사용될 수 있습니다:

- 솔루션 내에서 제공되는 데이터 컨트롤 및 UI 컨트롤의 정렬, 필터를 지정할 때
- 컨트롤의 속성 또는 스타일을 데이터에 따라 동적으로 결정하기 위해 표현식 바인딩을 사용할 때
- 일반 JavaScript 배열을 필터링 하거나, 소트 할 때
- 일반 JavaScript 객체를 입력으로 하여 특정한 값을 계산할 때

#### 7.3.2.1. 컨트롤에 표현식 추가

컨트롤의 바인딩 컨텍스트를 참조하여 데이터셋의 로우 정보를 가지고 데이터를 출력할 수 있습니다.

아래는 인풋박스 컨트롤에 enabled 속성에 익스프레션 바인딩을 하여, 그리드가 추가된 경우 비활성화 처리를 하는 표현식을 작성한 예시입니다.



Binding Property dialog box showing the configuration for the 'getStateString()' property binding.

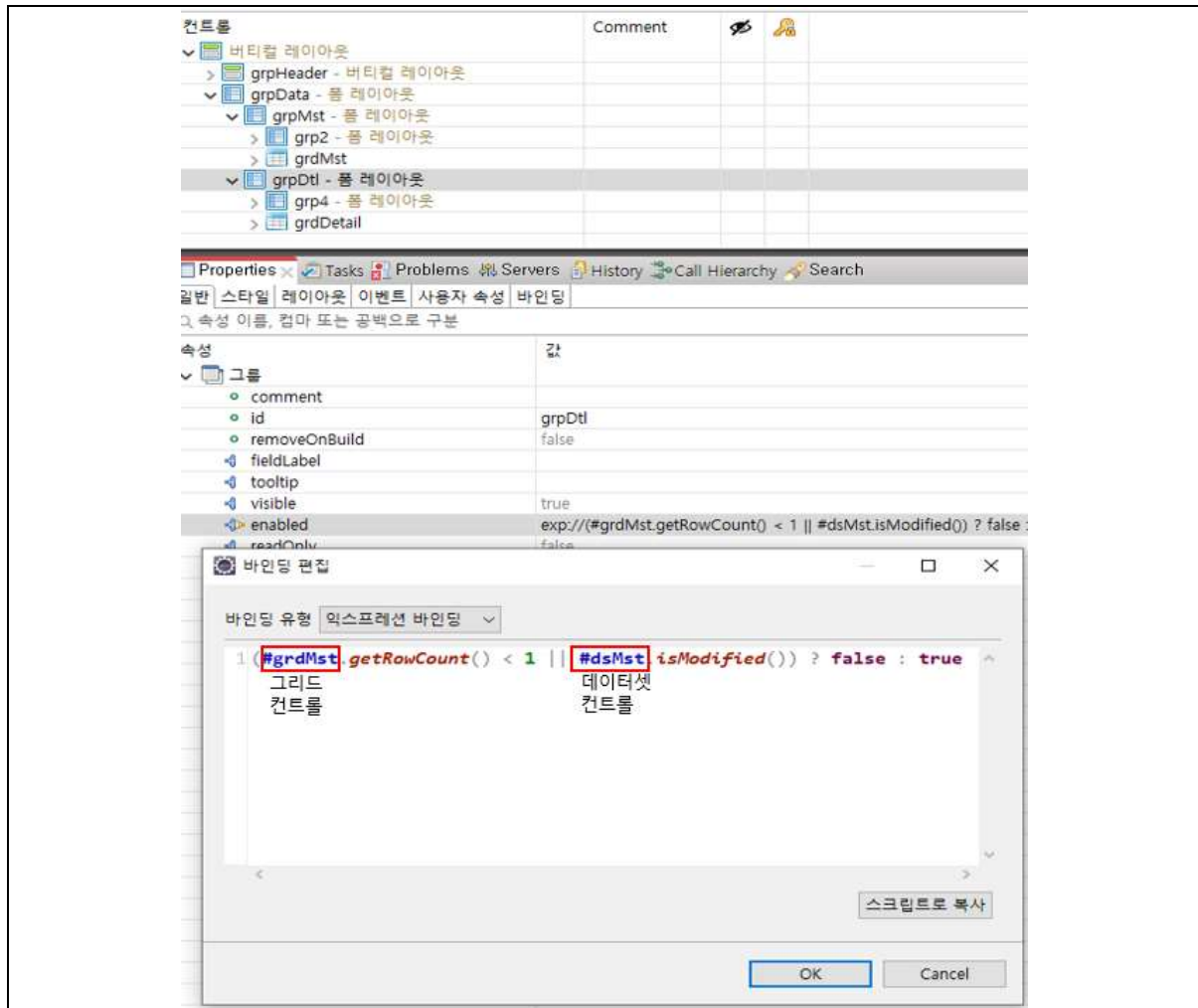
ID	Task	Start	Progress	note
id1	Caranville Drive	2018-05-10	0%	note1
id2	Buy a milk	2018-05-01	100%	note2
id3	Trip to Japan	2018-04-21	50%	note3
id4	Book a concert ticket	2018-03-01	40%	note4

ID	Task	Start	Progress	note
id1	Caranville Drive	2018-05-10	0%	note1

ID	Task	Start	Progress	note
id1	Caranville Drive	2018-05-10	0%	note1
id2	Buy a milk	2018-05-01	100%	note2
id3	Trip to Japan	2018-04-21	50%	note3
id4	Book a concert ticket	2018-03-01	40%	note4

ID	Task	Start	Progress	note
id1	Caranville Drive	2018-05-10	0%	note1

아래는 그룹 컨트롤의 enabled 속성에 특정 그리드의 로우카운터와 특정 데이터셋의 변경 여부를 리턴하여 비활성화 처리를 하는 익스프레션 바인딩을 적용한 예시입니다.



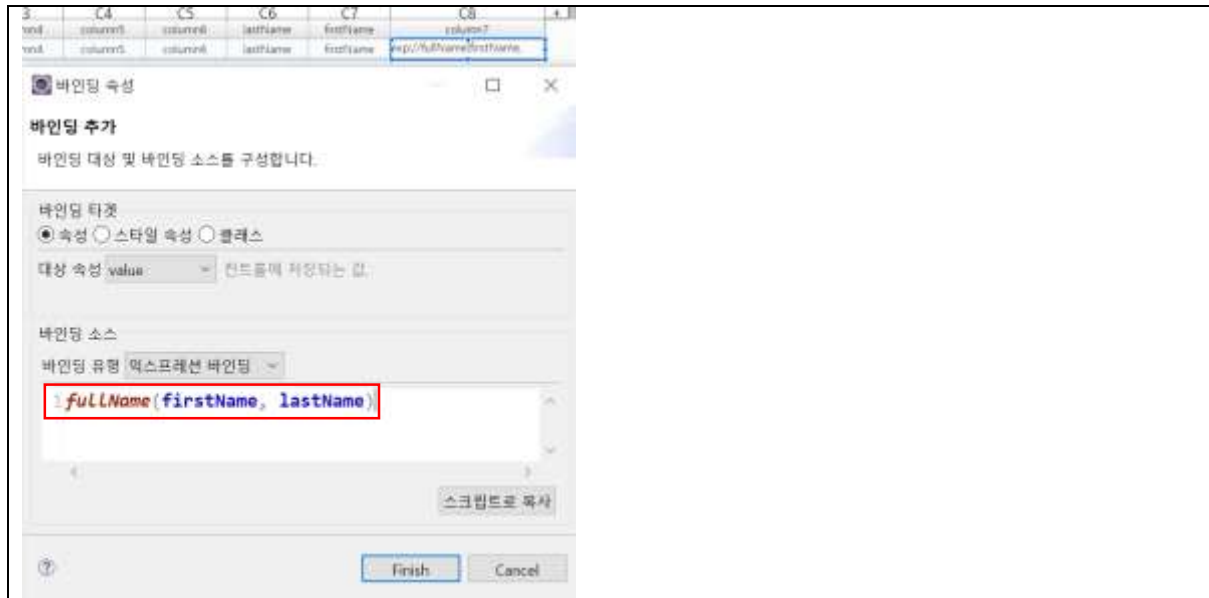
### 7.3.2.2. 사용자 정의 표현식

표현식은 기본적으로 연산자와 수학적 함수가 제공되며, 추가적으로 사용자가 함수를 등록할 수 있습니다. 아래와 같이 공통 모듈에 표현식에 사용할 함수를 작성하고 표현식 엔진에 등록합니다.

```
// 풀네임을 반환합니다.
function getFullName(firstName, lastName) {
    return firstName + " " + lastName;
}

// 표현식에서 사용할 수 있는 사용자 정의 함수를 등록합니다.
cpr.expression.ExpressionEngine.INSTANCE.registerFunction("fullName", getFullName);
```

그리드에 표현식 엔진에 등록한 함수를 사용할 컨트롤에 바인딩을 합니다. 파라미터로는 컬럼명을 지정합니다.



※ 그리드내에서 익스프레션 바인딩 혹은 표현식 엔진을 사용시 성능저하가 나타날 수 있습니다.

## 7.4. 스타일링

대부분의 UI 컨트롤들은 style 멤버 속성으로 접근할 수 있는 스타일러의 **css()** 메서드를 이용하여 원하는 CSS 속성을 줄 수 있습니다. 스타일러는 모든 css 속성의 바인딩 API 를 제공하며, 클래스를 변경하는 **addClass()** API 도 제공합니다. 특정 스타일러들은 하위 스타일러를 가질 수 있습니다. 예를 들어, 콤보박스의 스타일러는 콤보박스가 펼쳐졌을 때 나타나는 아이템들의 스타일을 정의하기 위해 **아이템 스타일러**를 제공합니다.

```
// 콤보박스 동적 생성
var cb = new cpr.controls.ComboBox();
// 클래스 추가
cb.style.addClass("test");
// 스타일 지정
cb.style.css("color", "red");
// 복수 스타일 지정
cb.style.css({
  "color" : "red",
  "background-color" : "orange"
});
// 스타일 열기
var color = cb.style.css("color");
// JSON 형태로 지정된 인라인 스타일 모두 열기
var styles = cb.style.css();
// 콤보 박스의 아이템 스타일 변경
cb.style.item.css("color", "red");
```

스타일러의 **css()** 메서드를 이용하여 스타일을 변경할 때, **컨트롤의 위치 및 크기는 부모 컨트롤과 부모 컨트롤의 레이아웃이 결정하기 때문에**, 부모 컨트롤과 레이아웃 컨스트RAINT를 통해 연결된 컨트롤들은 위치 및 크기에 영향을 주는 속성들(ex. left, top, right, bottom, width, height)을 변경하더라도 무시됩니다. 따라서, **컨트롤의 위치 및 크기를 변경하려면 레이아웃을 참고**하십시오.

## 8. 레이아웃

### 8.1. 레이아웃 유형

레이아웃	설명	컨스트레인트 속성	설명
XY 레이아웃	자식 컨트롤들을 XY 좌표체계를 이용하여 배치합니다.	horizontalAnchor	좌우 앵커 지정 값
		verticalAnchor	상하 앵커 지정 값
		top	부모의 위쪽 경계선과 떨어진 거리
		right	부모의 오른쪽 경계선과 떨어진 거리
		bottom	부모의 아래쪽 경계선과 떨어진 거리
		left	부모의 왼쪽 경계선과 떨어진 거리
		width	컨트롤의 너비
		height	컨트롤의 높이
반응형 XY 레이아웃	자식 컨트롤들을 별도 스크린 크기마다 다른 XY 좌표체계를 이용하여 배치하거나 숨깁니다.	hidden	현재 스크린에서 숨김 여부
		horizontalAnchor	좌우 앵커 지정 값
		verticalAnchor	상하 앵커 지정 값
		top	부모의 위쪽 경계선과 떨어진 거리
		right	부모의 오른쪽 경계선과 떨어진 거리
		bottom	부모의 아래쪽 경계선과 떨어진 거리
		left	부모의 왼쪽 경계선과 떨어진 거리
		width	컨트롤의 너비
		height	컨트롤의 높이
폼레이아웃	폼을 작성 할 때 사용하는 그리드 형식의 레이아웃입니다	horizontalAlign	컨트롤의 너비가 주어졌을 때 컨트롤의 가로 배치 방법
		verticalAlign	컨트롤의 높이가 주어졌을 때 컨트롤의 세로 배치 방법
		width	컨트롤의 너비
		height	컨트롤의 높이
		rowindex	로우 인덱스

		colIndex	컬럼 인덱스
		rowspan	로우 스팬 개수
		colspan	컬럼 스팬 개수
		ignoreLayoutSpacing	간격을 무시하고 가득 채울지 여부
버티컬 레이아웃	컨트롤을 세로로 배치할 수 있는 레이아웃입니다. 다양한 컨트롤들을 동적으로 배치해도 세로로 정렬이 되기 때문에 모바일 화면 개발 시 유용하게 사용할 수 있는 레이아웃입니다.	autoSize	자동 크기를 사용할 유형
		width	컨트롤의 너비
		height	컨트롤의 높이
		minWidth	자동 너비 적용시 사용할 최소 너비
플로우 레이아웃	컨트롤을 차례대로 배치할 수 있는 레이아웃입니다. 다양한 컨트롤들을 동적으로 배치해도 차례대로 배치가 되면서 그룹이 꽉 찰 경우 줄 바꿈이 자동으로 됩니다.	minHeight	자동 높이 적용시 사용할 최소 높이
		autoSize	자동 크기를 사용할 유형
		width	컨트롤의 너비
		height	컨트롤의 높이
		minWidth	자동 너비 적용시 사용할 최소 너비
		minHeight	자동 높이 적용시 사용할 최소 높이
		allowNewLine	컨트롤의 줄바꿈 허용 여부

## 8.2. 컨스트레인트 지정

한 컨트롤이 부모 컨트롤에 추가 될 때, 자식의 위치를 결정 짓는 정보를 레이아웃 컨스트레인트라고 합니다. 모든 자식 컨트롤들은 부모에 추가 될 때, 반드시 부모의 레이아웃과 일치하는 레이아웃 컨스트레인트를 지정해야 합니다.

### 8.2.1. XY 레이아웃

그룹에 **addChild()** API 를 사용하여 top, left, right, bottom 에 대한 앵커를 지정하고, width 와 height 를 지정합니다. 그룹의 레이아웃에 따라 constraint 안에 콘텐츠 어시스트 시 알맞은 속성값이 나타납니다. addChild() API 는 컨트롤을 추가한 순서대로 인덱스가 지정됩니다.

#### 그룹(컨테이너)에 XY 레이아웃 생성 및 지정

```
var xyLayout = new cpr.controls.layouts.XYLayout();
var xyGroup = app.lookup("grp1");
xyGroup.setLayout(xyLayout);
```

### XY 레이아웃이 설정된 그룹에 컨트롤 추가

```
var vcBtn = new cpr.controls.Button();
vcBtn.value = "버튼1";
// xyLayout 그룹에 버튼 컨트롤 추가
xyLayout.addChild(vcBtn, {
    top : "20px",
    left : "20px",
    width : "100px",
    height : "30px"
});
```

그룹 안에서 특정 컨트롤의 위치를 변경할 경우 **updateConstraint()** API 를 사용합니다.

### XY 레이아웃에서 컨트롤 위치 변경

```
// xyLayout 그룹에 버튼 컨트롤 위치 변경
xyLayout.updateConstraint(vcBtn, {
    top : "20px",
    left : "20px",
    height : "30px",
    width : "100px"
});
```

그룹 안에서 특정 인덱스에 컨트롤을 추가해야 할 경우 **insertChild()** API 를 사용합니다. 인덱스에 따라 탭 인덱스가 지정되며 그룹 안에 첫번째 자식 컨트롤을 얻고 싶을 경우 **getFirstChild()** API 를 사용하고, 마지막 자식 컨트롤을 얻고 싶을 경우 **getLastChild()** API 를 사용합니다. 그룹 안에 모든 컨트롤들을 얻고 싶을 경우 **getChildren()** API 를 사용합니다.

### XY 레이아웃에서 컨트롤의 인덱스 변경

```
var vcBtn = new cpr.controls.Button();
vcBtn.value = "버튼1";
// grp1 그룹에 첫번째 자식으로 버튼 컨트롤 추가
xyGroup.insertChild(0, vcBtn, {
    top : "20px",
    left : "150px",
    width : "300px",
    height : "30px"
});
```



### 8.2.2. 반응형 XY 레이아웃

반응형 XY 레이아웃 그룹에 컨트롤을 추가하기 위해서는 반드시 스크린에 지정된 미디어 쿼리를 작성해 주어야합니다. Hidden 속성에 true 로 지정할 경우 해당 미디어 쿼리에서 컨트롤은 숨겨집니다. 그 외에 속성들은 XY 레이아웃과 동일하게 앵커를 잡아주고 width 또는 height 를 작성합니다. 미디어 쿼리는 디자인 뷰 → 마우스 우클릭 → 보기 → 컴파일된 결과 파일 보기(배포 경로에 있는 clx.js 파일)에서 app.supportMedia()를 참고합니다.

#### 그룹(컨테이너)에 반응형 XY 레이아웃 생성 및 지정

```
var responsiveXYLayout = new cpr.controls.layouts.ResponsiveXYLayout()
var resXyGroup = app.lookup("grp1");
resXyGroup.setLayout(responsiveXYLayout);
```

#### 반응형 XY 레이아웃이 설정된 그룹에 컨트롤 추가

```
resXyGroup.addChild(new cpr.controls.Button("btn1"), {
    positions : [
        {
            // 스크린에 지정된 미디어 쿼리 default
            media : "all and (min-width: 1024px)",
            top : "20px",
            left : "20px",
            width : "300px",
            height : "90px"
        }, {
            // 스크린에 지정된 미디어 쿼리 tablet
            media : "all and (min-width: 500px) and (max-width: 1023px)",
            top : "20px",
            left : "20px",
            width : "200px",
            height : "60px",
            hidden : true // 숨김처리
        }, {
            // 스크린에 지정된 미디어 쿼리 mobile
            media : "all and (max-width: 499px)",
            top : "20px",
            left : "20px",
            width : "100px",
            height : "30px"
        }
    ]
});
```

반응형 XY 레이아웃 그룹에서 컨트롤에 위치를 변경시킬 때에도 동일하게 반드시 **미디어 쿼리**를 지정해 주어야 합니다.

#### 반응형 XY 레이아웃에서 컨트롤 위치 변경

```
resXyGroup.updateConstraint(app.lookup("btn1"), {
    positions : [
        {
            media : "all and (min-width: 1024px)",
            top : "",
            left : "",
            width : "300px",
            height : "90px",
            bottom : "20px",
            right : "20px"
        }
    ]
});
```

### 8.2.3. 폼 레이아웃

폼 레이아웃이 지정된 그룹에 컨트롤을 추가할 때에는 반드시 **colIndex** 와 **rowIndex** 를 지정해주어야 합니다. 폼 레이아웃은 테이블 형태로 되어 있어서 행과 열을 가지고 있습니다.

#### 그룹(컨테이너)에 폼 레이아웃 생성 및 지정

```
// Layout
var formLayout = new cpr.controls.layouts.FormLayout();
formLayout.horizontalSeparatorWidth = 1;
formLayout.verticalSeparatorWidth = 1;
formLayout.setColumns(["100px", "1fr"]);
formLayout.setUseColumnShade(0, true);
formLayout.setRows(["25px", "1fr"]);
var formGroup = app.lookup("grp1");
formGroup.style.setClasses(["cl-form-group"]);
formGroup.setLayout(formLayout);
```

#### 폼 레이아웃이 설정된 그룹에 컨트롤 추가

```
// 폼레이아웃 그룹에 첫번째 열, 첫번째 행에 버튼 컨트롤 추가
formGroup.addChild(new cpr.controls.Button("button1"), {
    colIndex : 0,
    rowIndex : 0
});
```

폼 레이아웃 그룹에서 컨트롤의 위치를 변경할 때에도 반드시 **colIndex** 와 **rowIndex** 를 지정해야 합니다.

#### 폼 레이아웃에서 컨트롤 위치 변경

```
// 폼레이아웃 그룹에 첫번째 열, 두번째 행에 버튼 컨트롤 이동
formGroup.updateConstraint(app.lookup("button1"), {
    colIndex : 0 ,
    rowIndex : 1
});
```

폼 레이아웃은 테이블 형태로 되어 있어서 행 또는 열을 숨기거나 보이는 동작을 지정할 수 있습니다.

#### 폼 레이아웃의 행 또는 열 숨김 처리

```
formGroup.getLayout().setRowVisible(1, false); // 두번째 행 숨김
formGroup.getLayout().setColumnVisible(1, false); // 두번째 열 숨김
```

### 8.2.4. 버티컬 레이아웃

버티컬 레이아웃이 지정된 그룹에 컨트롤이 추가될 때에는 세로 정렬되는 레이아웃이기 때문에 위에서부터 차례대로 컨트롤이 추가됩니다. 자동으로 크기를 조정하는 **autoSize** 속성을 조정할 수 있습니다. 컨트롤과 컨트롤 사이에 추가할 경우 **insertChild()** API 를 사용합니다.

#### 그룹(컨테이너)에 버티컬 레이아웃 생성 및 지정

```
var verticalLayout = new cpr.controls.layouts.VerticalLayout();
var vGroup = app.lookup("grp1");
vGroup.setLayout(verticalLayout);
```

#### 버티컬 레이아웃이 설정된 그룹에 컨트롤 추가

```
vGroup.addChild(new cpr.controls.Button("btn1"), {
    height : "30px",
    width : "100px",
    autoSize : "none"
});
```

버티컬 레이아웃이 지정된 그룹에 컨트롤 위치를 변경할 때에는 변경하고자 하는 **Constraints** 속성을 입력합니다.

#### 버티컬 레이아웃에서 컨트롤 위치 변경

```
vGroup.updateConstraint(app.lookup("btn1"), {
    width : "200px"
});
```

그룹 안에서 **특정 인덱스**에 컨트롤을 추가해야 할 경우 **insertChild()** API 를 사용합니다. 인덱스에 따라 탭 인덱스가 지정되며 그룹 안에 첫번째 자식 컨트롤을 얻고 싶을 경우 **getFirstChild()** API 를 사용하고, 마지막 자식 컨트롤을 얻고 싶을 경우 **getLastChild()** API 를 사용합니다. 그룹 안에 모든 컨트롤들을 얻고 싶을 경우 **getChildren()** API 를 사용합니다.

#### 버티컬 레이아웃에서 컨트롤의 인덱스 변경

```
// grp1 그룹에 첫번째 자식으로 버튼2 컨트롤 추가
vGroup.insertChild(0, new cpr.controls.Button("btn2"), {
    height : "30px",
    width : "100px",
    autoSize : "none"
});
```

#### 8.2.5. 플로우 레이아웃

플로우 레이아웃이 지정된 그룹에 컨트롤이 추가될 때에는 차례대로 정렬되는 레이아웃이기 때문에 **좌측에서부터 차례대로 컨트롤이 추가됩니다**. 자동으로 크기를 조정하는 **autoSize** 속성을 조정할 수 있습니다. 컨트롤과 컨트롤 사이에 추가할 경우 **insertChild()** API 를 사용합니다.

#### 그룹(컨테이너)에 플로우 레이아웃 생성 및 지정

```
var flowLayout = new cpr.controls.layouts.FlowLayout();
var fGroup = app.lookup("grp1");
fGroup.setLayout(flowLayout);
```

#### 플로우 레이아웃이 설정된 그룹에 컨트롤 추가

```
fGroup.addChild(new cpr.controls.Button("button1"), {
    height : "30px",
    width : "100px"
});
```

플로우 레이아웃이 지정된 그룹에 컨트롤 위치를 변경할 때에는 변경하고자 하는 **Constraints** 속성을 입력합니다.

#### 플로우 레이아웃에서 컨트롤 위치 변경

```
fGroup.updateConstraint(app.lookup("button1"), {
    height : "50px",
    width : "150px"
});
```

그룹 안에서 **특정 인덱스**에 컨트롤을 추가해야 할 경우 **insertChild()** API 를 사용합니다. 인덱스에 따라 탭 인덱스가 지정되며 그룹 안에 첫번째 자식 컨트롤을 얻고 싶을 경우 **getFirstChild()** API

를 사용하고, 마지막 자식 컨트롤을 얻고 싶을 경우 `getLastChild()` API 를 사용합니다. 그룹 안에 모든 컨트롤들을 얻고 싶을 경우 `getChildren()` API 를 사용합니다.

#### 플로우 레이아웃에서 컨트롤의 인덱스 변경

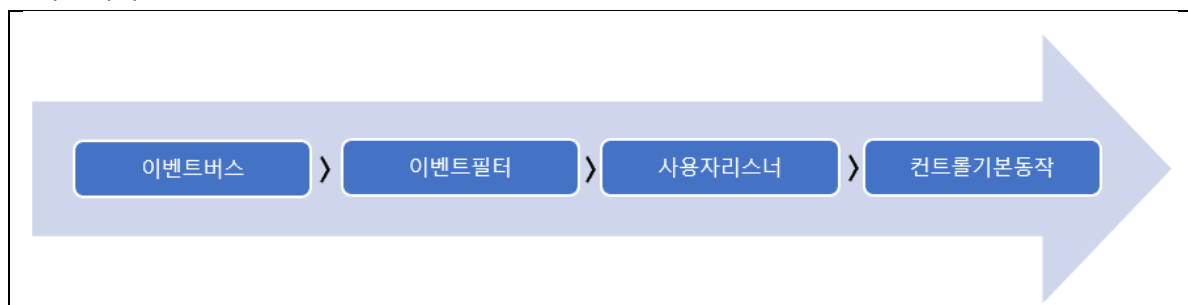
```
// grp1 그룹에 첫번째 자식으로 버튼2 컨트롤 추가
fGroup.insertChild(0, new cpr.controls.Button("btn2"), {
    height : "30px",
    width : "100px",
    autoSize : "none"
});
```

## 9. 이벤트

모든 이벤트는 **이벤트 버스**라고 불리는 **중앙 제어시스템**을 거쳐 전파가 이루어지며, 이를 통해 모든 이벤트를 감시하거나 차단할 수 있습니다. 이벤트 버스는 `cpr.events.EventBus.INSTANCE` 로 접근할 수 있습니다.

### 9.1. 이벤트 순서

eXBuilder6 에서 발생하는 모든 이벤트들은 다음의 과정을 거쳐 사용자가 작성한 리스너까지 전파됩니다.



모든 이벤트는 최초 이벤트 버스에서 처리되며, 필터링 과정을 거치게 됩니다. 필터를 통과한 이벤트들은 사용자들이 작성한 리스너에게 전달되며, 마지막으로 해당 이벤트에 대한 컨트롤의 기본 동작이 일어나게 됩니다.

#### 9.1.1. 컨트롤의 기본 동작 차단

사용자가 작성한 리스너에서 넘겨받은 이벤트 객체의 **`preventDefault()`**를 호출하면 컨트롤의 기본동작을 차단할 수 있습니다. 예를 들어, 그리드의 `before-selection-change` 이벤트에서 기본동작을 차단하면 `selection-change` 이벤트가 실행되지 않습니다. 하지만 모든 이벤트의 기본동작을 차단하는 것은 아니기 때문에 도움말에서 각 컨트롤의 이벤트 문서를 참고하시기 바랍니다.

## 9.2. 이벤트 버스 및 필터

모든 이벤트는 최초에 이벤트 버스에서 처리되며 특정 이벤트에 대해 필터를 추가할 수 있는 기능을 제공합니다. 이러한 목적의 이벤트 리스너를 이벤트 필터라고 합니다. 이벤트 필터는 해당 이벤트 발생시 공통적인 추가 작업을 하거나 조건에 따라 이벤트 전파 자체를 차단하는 목적으로 사용할 수 있습니다.

아래의 예시는 공통 모듈에 앱인스턴스에서 init 이벤트가 발생할 때마다, 앱에 포함되어 있는 모든 버튼에 클래스로 btn-default 를 추가하는 스크립트입니다. 이러한 **이벤트 버스 코드**는 반드시 공통 모듈 내에서 작성되는 것이 바람직합니다.

```
cpr.events.EventBus.INSTANCE.addFilter("contextmenu", function(e) {  
    if (e.control.type == "grid") {  
        e.preventDefault();  
    }  
});
```

이벤트 필터를 통과한 후 사용자가 작성한 리스너가 동작합니다.

```

/*
 * 그리드에서 contextmenu 이벤트 발생 시 호출.
 * 마우스의 오른쪽 버튼이 클릭되거나 컨텍스트 메뉴 키가 눌러지면 호출되는 이벤트.
 */
function onGrd1Contextmenu(/* cpr.events.CMouseEvent */ e){
    /**
     * @type cpr.controls.Grid
     */
    var grd1 = e.control;

    if(e.targetObject.relativeTargetName == "header"){
        /* 기존 우클릭메뉴 실행하지 않음 */
        e.preventDefault();

        /* 컨텍스트 메뉴 생성 */
        var vcContextMenu = new cpr.controls.Menu("grdHeaderMenu");
        vcContextMenu.addItem(new cpr.controls.MenuItem("오름차순", "value1", null));
        vcContextMenu.addItem(new cpr.controls.MenuItem("내림차순", "value2", null));

        /* 컨텍스트 메뉴의 위치를 지정 */
        var voAppRect = app.getActualRect();

        vcContextMenu.style.css({
            position: "absolute",
            top: "" + (e.clientY - voAppRect.top) + "px",
            left: "" + (e.clientX - voAppRect.left) + "px",
            width: "200px"
        });
        /* 앱 위에 컨텍스트 메뉴 배치 */
        app.floatControl(vcContextMenu);

        /* floatControl API를 통해 배치된 메뉴에 포커스 지정 */
        vcContextMenu.focus();

        /* 메뉴 컨트롤에 blur 이벤트 지정 이후, blur 될 때 dispose API를 통해 메뉴 객체 삭제 */
        vcContextMenu.addEventListener("blur", function(e) {
            vcContextMenu.dispose();
        });
        var vnCellIdx = e.targetObject.cellIndex;

        /* 메뉴 컨트롤에 item-click 이벤트 지정 이후, item-click 될 때 해당 컬럼 정렬 */
        vcContextMenu.addEventListener("item-click", function(e){
            if (e.item.value == "value1") {
                grd1.header.sort([{cellIndex : vnCellIdx, asc : true}]);
            } else {
                grd1.header.sort([{cellIndex : vnCellIdx, asc : false}]);
            }
            vcContextMenu.dispose();
        });
    }
}

```

조건에 따라 이벤트의 전파를 중단시킬 수도 있습니다.

```

cpr.events.EventBus.INSTANCE.addFilter("click", function(e) {
    if (shoudDispatch(e) == false) {
        e.stopPropagation();
    }
});

```

### 9.3. 알림

알림을 표현하는 컨트롤입니다. 사용자의 특정 상태나 동작에 따라 화면에 해당 내용을 전달하고자 할 때 알림 컨트롤을 사용합니다.

NotificationCenter(글로벌 메시지 큐)를 이용하여 해당 주제를 구독하고 있는 곳에 메시지를 송신할 수 있습니다.

#### 수신(Subscribe)

```
cpr.core.NotificationCenter.INSTANCE.subscribe("app-msg", app, function(poMsgInfo) {
    var vcNotiToastr = app.lookup("notiToastr");
    //메시지의 타입이 info 인 경우
    if(poMsgInfo["type"] == "info"){
        vcNotiToastr.infoDelay = 2000;
        vcNotiToastr.info(poMsgInfo["msg"]);
    }
});
```

#### 송신(Post)

```
/* 알림 보내는 곳 (Main.clx에 존재) */
cpr.core.NotificationCenter.INSTANCE.post("app-msg", {
    msg : "알림 챗터에서 보내는 메세지입니다.",
    type : "info"
});
```



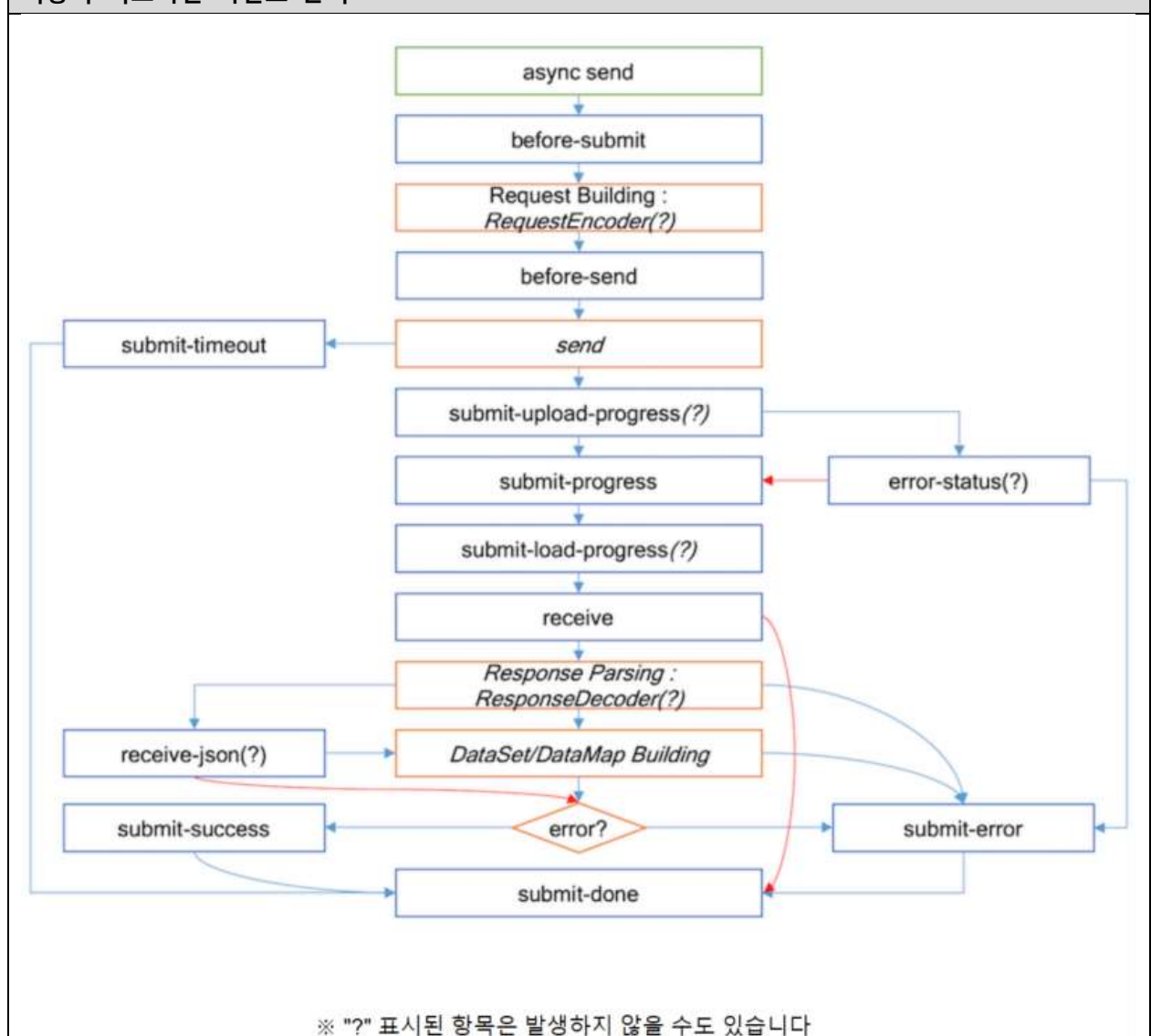
## 10. 서브미션

eXBuilder6 앱에서 서버와 통신을 처리하기 위해 데이터(데이터 셋, 맵)를 전달하고 서버로부터 가공된 데이터를 받아 관리하는 컨트롤입니다.

서브미션은 HTTP 또는 HTTPS 프로토콜을 통해 서버에 데이터를 전달하고 서버로부터 처리 결과 데이터를 수신합니다. 서브미션을 이용하여 App 은 HTTP Header, 데이터맵, 데이터셋 그리고 추가 Parameter 를 전달할 수 있으며 데이터맵, 데이터셋, 파일, JavaScript 그리고 부가적인 메타 정보를 수신할 수 있습니다.

### 10.1. 서브미션 이벤트 순서

비동기 서브미션 이벤트 순서



비동기 서브미션은 위와 같은 형태의 이벤트 순서를 가지고 있습니다. 실제 우리가 **send()** API 를 통해 서브미션을 전송하고 데이터가 넘어오는 것은 **receive()** 이벤트를 통해서 확인할 수 있으며 그 이후 데이터 셋 혹은 데이터 맵에 데이터가 들어오고 submit-done 이후에 데이터들이 화면에

보여지게 됩니다. **submit-done** 이벤트는 통신 오류 또는 성공 여부에 상관없이 항상 마지막에 발생합니다.

이벤트	설명
<b>before-submit</b>	통신을 시작하기 전에 발생합니다.
<b>before-send</b>	XMLHttpRequest가 open된 후 send 함수가 호출되기 직전에 발생합니다. 타 솔루션과 연동을 위해 제공되는 이벤트로 Submission.xhr을 획득하여 헤더를 추가하는 것 이외에 다른 작업은 오동작을 발생시킬 수 있어 주의가 필요합니다.
<b>submit-upload-progress</b>	multipart/form-data일 때 서버로 일정 크기의 데이터를 전송했을 때 발생합니다.
<b>submit-progress</b>	서버로부터 일정 크기의 데이터를 전송 받았을 때 발생합니다. 하나의 응답에 대해 발생하지 않거나 여러 번 발생 할 수 있습니다. 비동기로 동작할 때만 사용할 수 있습니다.
<b>submit-load-progress</b>	서버로부터 수신된 데이터가 응답 데이터 컨트롤에 부분적으로 적재되었을 때 (TSV 프로토콜의 경우 부분적으로 적재되었을 때) 발생합니다. 하나의 응답에 대해 여러 번 발생할 수 있습니다.
<b>receive</b>	서버로부터 데이터를 모두 전송 받았을 때 발생합니다.
<b>receive-json</b>	응답 프로토콜이 json일 때 서버로부터 받은 JSON 문자열을 JSONObject로 파싱에 성공했을 때 발생합니다.
<b>submit-success</b>	서버로부터 데이터를 모두 전송받고, 서브미션의 응답데이터의 데이터셋, 데이터맵에 처리가 끝난 후에 발생합니다.
<b>error-status</b>	서브미션이 전송된 후 수신받은 서버의 응답상태코드가 200이 아닐 때 발생합니다. error-status 이벤트 핸들러에서 이벤트의 preventDefault 함수를 호출하면 서버의 응답메세지를 모두 수신한 후 submit-error 이벤트를 발생시킵니다. preventDefault 함수를 호출하지 않으면 서버의 응답메세지를 수신하지 않고 즉시 submit-error 이벤트를 발생시킵니다. 비동기로 동작할 때만 사용할 수 있습니다.
<b>submit-error</b>	통신에 오류 발생시 호출됩니다.
<b>submit-timeout</b>	통신 중 Timeout이 발생했을 때 호출되는 이벤트입니다. 비동기로 동작할 때만 사용할 수 있습니다.
<b>submit-done</b>	통신의 오류 또는 성공 여부에 상관 없이 마지막에 항상 발생합니다.

## 10.2. 서브미션 속성 설명

이름	설명
<b>async</b>	활성화되어 있을 경우 데이터 전송을 비동기 방식으로 수행하도록 설정합니다.
<b>withCredentials</b>	Cross Domain 요청시 자격증명 정보의 전송 여부를 설정합니다.
<b>action</b>	데이터를 전송할 서버의 주소를 설정합니다.
<b>method</b>	<p>데이터의 전송 방식을 설정합니다.</p> <ul style="list-style-type: none"> <li>• get : URL 에 변수를 포함시켜 요청합니다. 브라우저에 따라 길이제한이 있습니다.</li> <li>• post : URL 에 데이터가 노출되지 않습니다. 길이 제한이 없습니다. (default)</li> <li>• patch : 요청된 데이터의 일부를 수정합니다.</li> <li>• put : 요청된 데이터 전체를 수정합니다.</li> <li>• options : 웹 서버에서 지원되는 메서드의 종류를 확인합니다.</li> <li>• head : GET 방식과 동일하지만, 응답에 body 가 없고 응답코드와 head 만 응답합니다.</li> <li>• delete : 요청된 데이터 삭제를 요청합니다.</li> </ul>
<b>mediaType</b>	<p>전송 데이터의 미디어 형식을 지정합니다.</p> <ul style="list-style-type: none"> <li>• application/json : JSON 타입으로 요청합니다.</li> <li>• application/json;massdata : 대량 데이터를 JSON 타입으로 서버에 전송할 수 있는 형식입니다.</li> <li>• application/x-www-form-urlencoded : 기본 전송 형식입니다. (default)</li> <li>• application/x-www-form-urlencoded;simple : 파라미터 명에 alias 가 제거된 형식입니다.</li> <li>• application/x-www-form-urlencoded;massdata : 대량 데이터를 기본 전송 형식으로 서버에 전송할 수 있는 형식입니다.</li> <li>• multipart/form-data;encoding=json : JSON 타입으로 인코딩합니다.</li> <li>• multipart/form-data : 파일이 포함된 경우 사용합니다.</li> <li>• multipart/form-data;simple : alias 가 제거된 형식입니다.</li> </ul>
<b>responseType</b>	<p>서버가 요청에 대해 응답할 때 사용하는 데이터 형식을 지정합니다.</p> <ul style="list-style-type: none"> <li>• text : eXBuilder6 의 기본 형식을 받습니다. (default)</li> <li>• javascript : 자바 스크립트 형식으로 받습니다.</li> <li>• blob : 파일의 데이터를 받아 다운로드 합니다.</li> <li>• filedownload : 대용량 파일을 전송받을 때 사용합니다.</li> </ul> <p>서브미션의 receive, success, done 이벤트가 발생하지 않습니다.</p>

mediaType 별 허용가능한 HTTP method	
mediaType	AvailabledMethods
application/x-www-form-urlencoded	post, get, delete, head, options, patch, put
application/x-www-form-urlencoded;massdata	post, patch, put
application/x-www-form-urlencoded;simple	post, get, delete, head, options, patch, put
application/json	post, patch, put
application/json;massdata	post, patch, put
multipart/form-data	post, patch, put
multipart/form-data;encoding=json	post, patch, put
multipart/form-data;simple	post, patch, put

### 10.3. 데이터 형식 변환

eXBuilder6 프로토콜 관련 서버플러그인을 사용하지 않고 서버로 전송하는 데이터 형식이 기본 양식의 JSON 형태와 다른 경우 상황에 맞게 데이터를 변환 가능하게 도와주는 핸들러를 제공합니다.

Encoding
<p>서버로 전송하는 데이터를 별도의 형식으로 Encoding할 수 있는 함수를 등록합니다.</p> <pre>/**  * @param {cpr.protocols.Submission} submission  * @param {Object} reqData  */ function _requestEncoder(submission, reqData) {     var _app = submission.getAppInstance();     var reqDataObj = reqData["data"];     var paramObj = reqData["param"];      var voJsonType = {};     for (var key in reqDataObj) {         voJsonType[key] = reqDataObj[key];     }     return {"content" : voJsonType}; }</pre>
Decoding
<p>서버로부터 전송받은 eXBuilder6에서 지원하지 않는 형식의 데이터를 eXBuilder6에서 지원하는 형식으로 변환하거나 자체적으로 처리할 수 있는 Decoding 함수를 등록합니다.</p> <p>ResponseDecoder가 등록된 Submission은 "submit-load-progress" 이벤트를 발생시키지 않습니다.</p> <pre>/**  * @private  * @param {cpr.protocols.Submission} submission</pre>

```

* @param {Object} reqData
*/
function _responseDecoder(submission, resData) {
    var _app = submission.getAppInstance();
    var resDataObj = JSON.parse(resData);

    var voProtocolJson = {};

    for (var key in resDataObj) {
        for (var subKey in resDataObj[key]) {
            voProtocolJson[subKey] = resDataObj[key][subKey];
        }
    }

    return {contentType : "application/json", content : voProtocolJson};
}

```

#### 서브미션 before-submit 이벤트

```

/*
 * 서브미션에서 before-submit 이벤트 발생 시 호출.
 * 통신을 시작하기전에 발생합니다.
 */
function onSubmission1BeforeSubmit(/* cpr.events.CSubmissionEvent */ e){
    /**
     * @type cpr.protocols.Submission
     */
    var submission1 = e.control;
    //서브미션 보내기 전에 서브미션 타입이 json인 경우
    if(submission1.mediaType == "application/json"){
        submission1.setRequestEncoder(_requestEncoder);
        submission1.setResponseDecoder(_responseDecoder);
    }
}

```

## 10.4. 엑셀 export

### 10.4.1. 엑셀 & PDF export

그리드에 출력된 데이터를 서버로 전달하여 엑셀 내보내기 처리를 위한 API 를 제공합니다.  
그리드 스타일은 **인라인 스타일**만 적용되어 export 되고 관련 서버플러그인은 아파치 POI 라이브러리를 사용하므로 아래 최소 권장 사항을 확인해야 합니다.

<b>Excel export 기능 사용 시</b>
Apache POI 3.15 이상
<b>PDF export 기능 사용 시</b>
Apache PDFBox 2.0.11 이상

PDF export 사용 라이브러리

라이브러리	사이트
pdfbox	<a href="https://mvnrepository.com/artifact/org.apache.pdfbox/pdfbox">https://mvnrepository.com/artifact/org.apache.pdfbox/pdfbox</a>
fontbox	<a href="https://mvnrepository.com/artifact/org.apache.pdfbox/fontbox">https://mvnrepository.com/artifact/org.apache.pdfbox/fontbox</a>

### 엑셀로 export 하기 위한 getExportData API()

```
var exportData = app.lookup("grid").getExportData({
  exceptStyle : true, //셀 스타일을 Export시 제외할 지 여부. (default true)
  applyFormat : true, //Export시 포맷을 적용시킬 지 여부. number 데이터 타입을 엑스포트를 하기 위해서는 해당 옵션을 false로 주어야 합니다. (default true)
  applySuppress : true, //엑셀 엑스포트를 suppress, mergedToIndexExpr에 의한 셀 병합을 반영할지 여부. (default false)
  excludeCols : [], //Export시 제외시킬 header cellIndex 배열.
  freezeHeader : false, //엑셀 Export 시 해당 부분을 동고정할지 여부. (default false)
  rows : [], //Export시할 특정 row index 배열.
  useFormat : true, //엑셀 파일 (xlsx, xls 등)으로 엑스포트를 해당 파일이 format속성을 사용할지 여부. (default true)
  rowDataHandler: function(datas, rowIndex){ // 엑셀 Export 시 열별 스타일 또는 데이터를 변경할 수 있는 핸들러.
  }
});

//엑스포트를 완료한 타이머에서는 아래와 같은 metadata를 추가할 수 있습니다.
exportData["metadata"] = {};
//옵션을 설정합니다.
exportData["metadata"]["password"] = "1234";
//변세 방향을 설정합니다 "PORTRAIT" | "LANDSCAPE"
exportData["metadata"]["printPageOrientation"] = "PORTRAIT";
//기본 값은 false 이고 true로 설정된 경우 강제 개행문자가 있는 행은 wrap 됩니다.
exportData["metadata"]["wraplineBreak"] = true;

// 다운로드 받을 파일명을 설정합니다.
// 서버에서 해당 규칙으로 접근시 export 로직이 실행되도록 준비된 상태여야 합니다.
var submission = new cpr.protocols.Submission("sub_excelExport");
submission.action = "../export/" + app.lookup("grid").id + ".xlsx";
submission.mediaType = "application/json";
submission.responseType = "blob";
// 그리드 데이터를 request data로 설정합니다.
submission.setRequestObject(exportData);
// 데이터를 전송합니다.
submission.send();
```

실제 네트워크에서 확인할 수 있는 export 된 데이터

```

▼ {metadata: {...}, name: null, cols: Array(10), rowgroups: Array(3)}
  ▼ cols: Array(10)
    ▶ 0: {width: "25px"}
    ▶ 1: {width: "50px"}
    ▶ 2: {width: "100px"}
    ▶ 3: {width: "100px"}
    ▶ 4: {width: "100px"}
    ▶ 5: {width: "100px"}
    ▶ 6: {width: "100px"}
    ▶ 7: {width: "100px"}
    ▶ 8: {width: "100px"}
    ▶ 9: {width: "100px"}
    length: 10
    ▶ __proto__: Array(0)
  ▼ metadata:
    password: null
    printPageOrientation: null
    wrapLineBreak: false
    ▶ __proto__: Object
    name: null
  ▼ rowgroups: Array(3)
    ▶ 0: {region: "header", style: Array(10), data: Array(1)}
    ▶ 1: {region: "detail", style: Array(10), data: Array(12)}
    ▶ 2: {region: "footer", style: Array(20), data: Array(1)}
    length: 3
    ▶ __proto__: Array(0)
  ▶ __proto__: Object

```

### CleopatraFileExportController.java 컨트롤러

```

@RequestMapping("/export/{fileName}.pdf")
public void exportPDF(HttpServletRequest request, HttpServletResponse response) throws IOException {
    String downloadFileName = "test" + ".pdf";
    downloadFileName = this.encodingDownloadFileName(request, downloadFileName);

    response.setCharacterEncoding("utf-8");
    response.setContentType("application/pdf");
    response.addHeader("Content-Disposition", "attachment;filename=\"" + downloadFileName + "\"");

    this.export(request, response, downloadFileName, EXPORTTYPE.PDF);
}

@RequestMapping("/export/{fileName}.xls")
public void exportXLS(HttpServletRequest request, HttpServletResponse response, @PathVariable("fileName") String fileName) throws IOException {
    String downloadFileName = fileName + ".xls";
    downloadFileName = this.encodingDownloadFileName(request, downloadFileName);

    response.setContentType("application/vnd.ms-excel");
    response.addHeader("Content-Disposition", "attachment;filename=\"" + downloadFileName + "\"");

    this.export(request, response, fileName, EXPORTTYPE.XLS);
}

@RequestMapping("/export/{fileName}.xlsx")
public void exportXLSX(HttpServletRequest request, HttpServletResponse response, @PathVariable("fileName") String fileName) throws IOException {
    String downloadFileName = fileName + ".xlsx";
    downloadFileName = this.encodingDownloadFileName(request, downloadFileName);

    response.setContentType("application/vnd.openxmlformats-officedocument.spreadsheetml.sheet");
    response.addHeader("Content-Disposition", "attachment;filename=\"" + downloadFileName + "\"");

    this.export(request, response, fileName, EXPORTTYPE.XLSX);
}

```



```

private void export(HttpServletRequest request, HttpServletResponse response, String fileName, EXPORTTYPE type) throws IOException {
    response.setCharacterEncoding("utf-8");
    String newFileName = URLEncoder.decode(fileName, "utf-8");
    DataSource dataSource = JSONDataSourceBuilder.build(request, newFileName);
    OutputTarget outputTarget = new HttpServletResponseOutputTarget(response);

    ExporterFactory exporterFactory = ExporterFactory.getInstance();
    Exporter exporter = exporterFactory.getExporter(type);

    if(type == EXPORTTYPE.CSV){
        ((CSVExporter)exporter).setAutoWrap(false);
    }

    exporter.export(dataSource, outputTarget);

    response.flushBuffer();
}

private String encodingDownloadFileName(HttpServletRequest request, String psDownloadFileName) throws UnsupportedEncodingException {
    String downloadFileName = psDownloadFileName;
    String userAgent = request.getHeader("User-Agent");

    if(userAgent.contains("MSIE") || userAgent.contains("Chrome") || (userAgent.contains("Windows") && userAgent.contains("Trident"))){
        downloadFileName = URLEncoder.encode(downloadFileName, "utf-8");
        downloadFileName = downloadFileName.replaceAll("\\\\+", "%20");
    }

    return downloadFileName;
}

```

## 10.5. 대용량 데이터

대용량 데이터의 수신은 응답헤더의 ContentType 를 **text/tab-separated-values** 로 설정하여 행 단위 데이터를 전송 시 시각화 처리를 지원합니다.

서브미션의 **submit-load-progress 이벤트**는 서버로부터 수신된 데이터가 응답 데이터 컨트롤에 부분적으로 적재되었을 때 발생하며 하나의 응답에 대해 여러 번 발생할 수 있습니다.

### 행 단위로 데이터 전송

```

@RequestMapping("/list.do")
public void list(HttpServletRequest request, HttpServletResponse response, DataRequest dataRequest) throws Exception {
    // 서버 코드에서는 시각화를 간략하기 위해 행단위로 데이터를 전송합니다.
    // exBuilder6 Submission에서 행단위 처리를 가능하게 하기 위해서 응답헤더의 ContentType을 text/tab-separated-values 로 설정합니다.
    // 이렇게 전송하면 Submission에서는 응답헤더의 ContentType이 text/tab-separated-values 이면 submit-progress 이벤트를 동작시킵니다.
    final DataResponse dataResponse = DataResponse.getInstance(TSVResponseBuilder.CONTENT_TYPE, response);

    for(int i = 1; i <= 1000000; i++) {
        Map rowData = new HashMap();

        rowData.put("column1", "대구분-" + i);
        rowData.put("column2", "중구분-" + i);
        rowData.put("column3", "소구분-" + i);
        rowData.put("column4", "세구분-" + i);
        rowData.put("column5", "십세구분-" + i);
        rowData.put("column6", RandomStringUtils.randomAlphabetic(5));
        rowData.put("column7", RandomStringUtils.randomAlphabetic(5));
        rowData.put("column8", RandomStringUtils.randomAlphabetic(5));
        rowData.put("column9", RandomStringUtils.randomAlphabetic(5));
        rowData.put("column10", RandomStringUtils.randomAlphabetic(5));

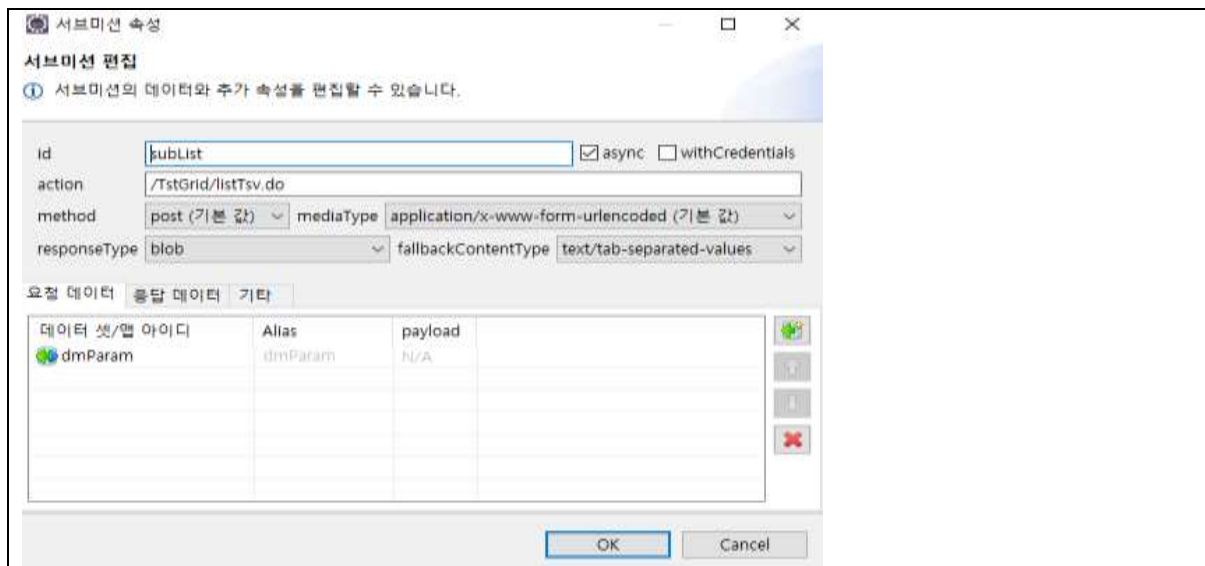
        try {
            dataResponse.send(rowData);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    dataResponse.flush();
}

```

서브미션 responseType blob, fallbackContentType text/tab-separated-values





### submit-load-progress 이벤트 처리 (전달받은 데이터 건수 출력)

```
/**
 * @param {cpr.events.CSubmissionEvent} e
 */
SubmissionKit.prototype._onSubmitLoadProgress = function(e) {
    /**
     * @type cpr.protocols.Submission
     */
    var submission = e.control;
    var loadmask = this._getLoadMask(submission);
    if(loadmask){
        try {
            if(submission.getResponseDataCount() > 0){
                var rowCnt = submission.getResponseData(0).data.getRowCount();
                loadmask.module.count(rowCnt);
            }
        } catch(ex){}
    }
};
```

## 10.6. 파일업로드/다운로드

eXBuilder6에서는 파일 업로드 및 다운로드가 가능한 파일업로드 컨트롤을 제공합니다.

파일업로드는 Submission의 mediaType을 multipart/form-data로 설정하여 처리합니다.

※ 도움말에 eXBuilder6 > 앱 개발 > 컨트롤 > 프로토콜 > 고급 > 파일 다운로드 / 파일 업로드를 참고하시기 바랍니다.

### 10.6.1. 파일업로드

파일 업로드 시 임시파일저장소로 업로드 되고 이 파일 정보를 스프링프레임워크의 MultipartResolver 나 com.cleopatra.protocol.data.DataRequest를 통해 파일 객체를 취득합니다.

파일인풋 컨트롤에 파일 추가 후 서브미션에 파일 추가

```
var vcFi = app.lookup("fi1");
app.lookup("subUpload").addFileParameter(vcFi.file.name, vcFi.file);
```

서브미션

**서브미션 속성**

서브미션 편집

① 서브미션의 데이터와 추가 속성을 편집할 수 있습니다.

id: subUpload ☒ async ☐ withCredentials

action: /CmnFile/upload.do

method: post (기본 값) **mediaType: multipart/form-data**

responseType: text fallbackContentType: application/json (?)

요청 데이터 응답 데이터 기타

데이터 셋/맵 마...	Alias	payl...
dmParam	dmParam	N/A

OK Cancel

### 서버 로직

```
@RequestMapping("/upload.do")
public void upload(HttpServletRequest request, HttpServletResponse response, DataRequest dataRequest) throws IOException {
    Map<String, UploadFile[]> uploadFiles = dataRequest.getUploadFiles();

    if(uploadFiles != null && uploadFiles.size() > 0) {
        Set<Entry<String, UploadFile[]>> entries = uploadFiles.entrySet();
        for(Entry<String, UploadFile[]> entry : entries) {
            UploadFile[] uFiles = entry.getValue();
            for(UploadFile ufile : uFiles){
                File file = ufile.getFile();
                //TODO 파일 저장 로직 작성
            }
        }
    }
}
```

스프링프레임워크에서 MultipartResolver 를 설정하였을 때 MultipartResolver 결과물을 DataRequest 로 받을 수 있는 파일업로드 핸들러를 지원합니다.

### exbuilder.json

```
{
  "fileupload": {
    "maxbodysize": -1.0,
    "maxfilesize": -1.0,
    "tempdirpath": "",
    "thresholdsize": 10240.0,
    "fileuploadhandler": null,
    "fileconverter": null,
    "fileuploadhandler": "com.cleopatra.protocol.parser.fileupload.CommonsFileUploadHandler"
  },
}
```

fileuploadhandler 는 com.cleopatra.protocol.parser.fileupload.FileUploadHandler 인터페이스를 구현한 클래스로 eXbuilder6 서버 플러그인에 내장된 핸들러는 다음과 같습니다.

클래스명	설명	비고
com.cleopatra.protocol.parser.fileupload.CommonsFileUploadHandler	별도로 핸들러를 등록하지 않으면 동작하는 기본 핸들러. Apache Commons Fileupload 라이브러리를 이용하여 파일업로드를 처리하며 exbuilder.json 파일에서 "maxbodysize", "maxfilesize", "tempdirpath", "thresholdsize" 설정을 통해 업로드 환경 설정가능.	Default. Apache Commons Fileupload 1.3.1 이상의 버전이 설치되어야 하고 플러그인 동작전에 파일업로드를 처리하는 동작이 없어야 함 (Springframework 의 multipartResolver 등이 먼저 동작하지 않아야 함).
com.cleopatra.protocol.parser.fileupload.Servlet3FileUploadHandler	Servlet3.0 파일업로드 API 를 이용하여 파일업로드를 처리.	Servlet 3.0 이상의 구동환경에서 동작하며 eXBuilder6 서버 플러그인이 실행되는 Servlet 에 Servlet Fileupload 가 동작할 수 있는 환경이 설정되어 있어야 함 (Servlet 3.0 API Guide 참조)
com.cleopatra.spring.fileupload.SpringMultipartResolverHandler	Springframework 환경에서 Springframework 의 multipartResolver 를 이용하여 파일업로드를 처리하는 핸들러.	Springframework 의 multipartResolver 가 설정되어 있고 정상동작해야 함

eXBuilder6 서버 플러그인에 의해 처리되는 multipart/form-data 처리를 위한 속성을 설정합니다.

속성명	설명
maxbodysize	HTTP POST Body 의 최대 용량(byte)이며, 생략될 경우 기본설정은 -1 로 무제한을 뜻합니다.
maxfilesize	업로드 되는 파일 개개의 최대 용량(byte)이며, 생략될 경우 기본설정은 -1 로 무제한을 뜻합니다.
tempdirpath	업로드된 파일을 옮길 임시 디렉토리 설정이며, 생략될 경우 기본 JAVA 임시 디렉토리가 설정됩니다.
thresholdsize	파일 업로드 처리시 최대 메모리 적재 용량을 설정(byte)이며, 생략될 경우 기본설정은 -1 로 무제한을 뜻합니다.
fileconvertor	eXBuilder6 서버 플러그인은 multipart/form-data 처리를 위해 Apache Commons 의 FileUpload 컴포넌트(1.3.1 이상)를 사용합니다. eXBuilder6 서버 플러그인은 업로드 된 파일을 java.io.File 객체로 관리하는데 이외에 별도의 객체로 관리해야 할 경우 org.apache.commons.fileupload.FileItem 를 관리할 객체로 변환하는 클래스를 설정하는데 사용하는 속성입니다. 제공되는 클래스 이외에 별도로 구현할 경우 com.cleopatra.protocol.parser.FileConvertor 를 구현해야 합니다. 설정은 구현된 클래스의 패키지명을 포함한 전체 이름을 등록합니다. 설정된 클래스에서 변환한 객체를 반환 받기 위해서는 com.cleopatra.protocol.data.DataRequest 의 getFileObjects 메소드를 통해 직접 얻어내거나 getUploadFiles 메소드를 통해 얻은 com.cleopatra.protocol.data.UploadFile 객체의 getFileObj 메소드를 통해

	<p>획득 가능합니다. 생략할 경우 com.cleopatra.protocol.parser.DefaultFileConvertor 가 동작하고 java.io.File 객체로 관리됩니다.</p> <p>Springframework 의 org.springframework.web.multipart.MultipartFile 로 관리가 필요한 경우 "com.cleopatra.spring.SpringFileConvertor" 로 설정할 수 있습니다.</p>
fileuploadhandler	<p>fileuploadhandler 는 com.cleopatra.protocol.parser.fileupload.FileUploadHandler 인터페이스를 구현한 클래스입니다.</p>

### 10.6.2. 파일 다운로드

서브미션을 통하여 서버의 파일을 다운로드할 수 있습니다. 파일 다운로드를 위해서 서브미션의 responseType 을 변경해야 합니다.

responseType	설명
<b>filedownload</b>	<ul style="list-style-type: none"> <li>• responseType 을 filedownload 로 설정할 경우, 파일 다운로드 처리를 브라우저가 담당하게 되므로 <b>대용량 파일의 다운로드</b> 시 적합합니다.</li> <li>• 설정된 mediaType 에 맞는 형식으로 요청 데이터가 생성되고 서버로 application/x-www-form-urlencoded 형식으로 전송됩니다.</li> <li>• 서브미션의 <b>submit-done</b> 이벤트는 요청 데이터를 <b>전송한 직후</b> 호출됩니다.</li> </ul>
<b>blob</b>	<ul style="list-style-type: none"> <li>• responseType 을 blob 으로 설정할 경우, 파일 다운로드 는 App 의 처리 프로세스와 통합되므로 <b>적은 용량의 파일 다운로드</b> 시 적합합니다.</li> <li>• 서버로부터 받은 응답 헤더 중 Content-Disposition 의 값으로 attachment 가 설정된 경우 파일다운로드 로 처리합니다.</li> <li>• 서브미션의 <b>submit-done</b> 이벤트는 서버로 파일다운로드가 <b>완료된 시점</b>에 호출됩니다.</li> </ul>

※ 적당한 파일 사이즈는 사용자 PC 환경에 따라 달라집니다.

## 서브미션 속성 창에서 responseType 을 filedownload 로 설정

**서브미션 속성**

서브미션 편집

① 서브미션의 데이터와 추가 속성을 편집할 수 있습니다.

id: subDownload ☒ async ☐ withCredentials

action: ../File/download.do

method: post (기본값) mediaType: application/x-www-form-urlencoded

responseType: **filedownload** fallbackContentType: application/json (?)

요청 데이터 응답 데이터 기타

데이터 셋/맵 아...	Alias	payl...
dmParamDown	dmParamDown	N/A

OK Cancel

## 다운로드 서버 예시

```

@RequestMapping("/download.do")
public void download(HttpServletRequest request, HttpServletResponse response, DataRequest dataRequest) throws IOException {

    ParameterGroup dmParamDown = dataRequest.getParameterGroup("dmParamDown");
    String fileName = dmParamDown.getValue("fileName");

    String resCharset = request.getHeader("res-charset");
    if ((resCharset == null) || (resCharset.equalsIgnoreCase(""))) {
        resCharset = "UTF-8";
    }

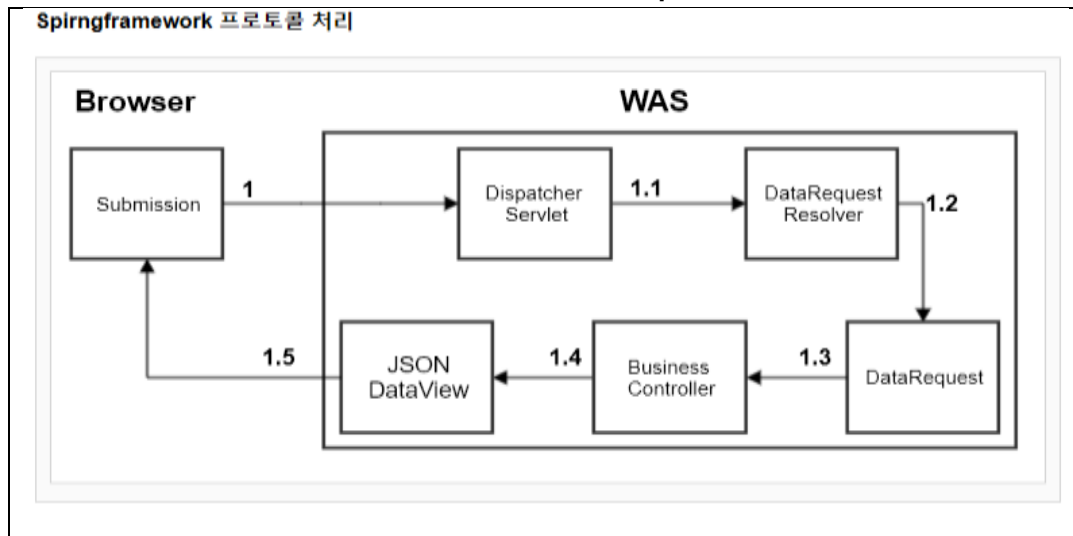
    List<Map<String, Object>> fileList = this.fileList();

    BufferedInputStream in = null;
    BufferedOutputStream out = null;
    try {
        response.setContentType("application/x-msdownload" + ";charset=" + resCharset);
        //유효성검 보편
        //외부에서 임의의 파일명 데이터 추가하는 경우에 HTTP 응답분할 취약점이 발생할 수 있으므로, \r, \n 문자를 제거한다.
        response.setHeader("Content-Disposition", "attachment;filename=\"" +
            fileName.replaceAll("[\\r\\n]", "") + "\";filename*=UTF-8'" + fileName.replaceAll("[\\r\\n]", "");
        in = new BufferedInputStream(new FileInputStream(String.valueOf(fileList.get(0).get("filePath"))));
        out = new BufferedOutputStream(response.getOutputStream());
        int data;
        while((data = in.read()) != -1){
            out.write(data);
        }
        out.flush();
    } catch (IOException e) {
        throw e;
    } catch (Exception e) {
        throw e;
    } finally {
        in.close();
        out.close();
    }
}

```

## 10.7. UI Adaptor(with springframework)

eXBuilder6 서버 플러그인은 HttpServletRequest 의 데이터를 수집하여 서브미션 요청 데이터를 서버 비즈니스 로직 처리에 편리하도록 파싱하여 **DataRequest** 에 적재합니다.



1. DispatcherServlet 는 서브미션이 요청한 데이터를 Business Controller 에 전달할 DataRequest 의 생성을 DataRequestResolver 에 요청합니다.
2. DataRequestResolver 는 HttpServletRequest 에서 추출한 데이터를 이용하여 DataRequest 를 생성합니다.
3. Business Controller 는 전달받은 DataRequest 를 이용하여 서브미션으로부터 전달된 요청 데이터를 사용할 수 있고 응답 데이터를 적재할 수 있습니다.
4. Business Controller 는 JSON 형식의 데이터를 브라우저로 전달하기 위해 JSONDataView 를 생성하여 리턴합니다.
5. JSONDataView 는 DataRequest 의 응답데이터를 JSON 형식으로 포매팅 한 후 서브미션에 전달합니다.

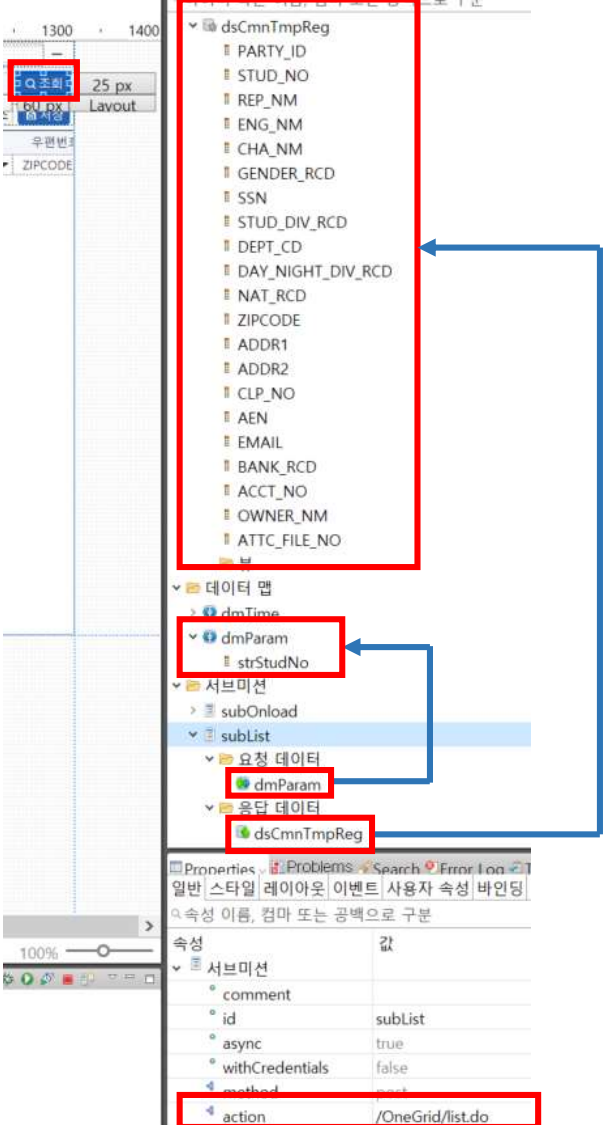
### ■ 관련 설정

파일명	내용
web.xml	<pre> &lt;listener&gt;   &lt;listener-class&gt;com.cleopatra.XBInitializer&lt;/listener-class&gt; &lt;/listener&gt;  &lt;servlet&gt;   &lt;servlet-name&gt;dispatcher&lt;/servlet-name&gt;   &lt;servlet-class&gt;org.springframework.web.servlet.DispatcherServlet   &lt;/servlet-class&gt;   &lt;init-param&gt;     &lt;param-name&gt;contextConfigLocation&lt;/param-name&gt;     &lt;param-value&gt;classpath:/config/context.xml&lt;/param-value&gt;   &lt;/init-param&gt;   &lt;load-on-startup&gt;1&lt;/load-on-startup&gt; &lt;/servlet&gt; &lt;servlet-mapping&gt;   &lt;servlet-name&gt;dispatcher&lt;/servlet-name&gt;   &lt;url-pattern&gt;*.clx&lt;/url-pattern&gt;   &lt;url-pattern&gt;*.do&lt;/url-pattern&gt; </pre>

	<pre> &lt;url-pattern&gt;*.csv&lt;/url-pattern&gt; &lt;url-pattern&gt;*.xls&lt;/url-pattern&gt; &lt;url-pattern&gt;*.xlsx&lt;/url-pattern&gt; &lt;/servlet-mapping&gt; </pre> <ul style="list-style-type: none"> <li>com.cleopatra.XBInitializer 는 eXBuilder6 서버 플러그인의 초기 설정을 처리하는 Listener입니다. (exbuilder.json, env.json 참조)</li> <li>dispatcher servlet-mapping 부분에 *.clx 패턴을 처리할 수 있도록 url-pattern을 추가합니다.(필수사항 아님)</li> </ul>
context.xml	<pre> &lt;bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"&gt;     &lt;!-- com.cleopatra.spring.DataRequestResolver 는 exbuilder6 프로토콜을 파싱하여 DataRequest 파라미터를 자동 생성해 주는 Resolver --&gt;     &lt;property name="customArgumentResolvers"&gt;         &lt;list&gt;             &lt;bean class="com.cleopatra.spring.DataRequestResolver"/&gt;         &lt;/list&gt;     &lt;/property&gt;     &lt;property name="webBindingInitializer"&gt;         &lt;bean class="egovframework.example.cmmn.web.EgovBindingInitializer"/&gt;     &lt;/property&gt; &lt;/bean&gt; </pre> <ul style="list-style-type: none"> <li>com.cleopatra.spring.DataRequestResolver 는 eXBuilder6 프로토콜을 파싱하여 DataRequest 파라미터를 자동 생성해 주는 Resolver입니다.</li> <li>multipartResolver설정은 생략해야 합니다.</li> </ul>



## ■ 참고 예

파일명	내용
oneGrid.clx	 <p>조회 이벤트 발생시 subList 서비스의 action경로로 해당 서블릿(컨트롤러)과 통신합니다. 이때 요청 데이터는 dmParam, 응답 데이터는 dsCmnTmpReg를 사용합니다.</p>



OneGridController.java

```

@RequestMapping("/list.do")
public View list(HttpServletRequest request, HttpServletResponse response, DataRequest dataRequest) throws Exception {

    //요청 데이터 그룹
    ParameterGroup dmParam = dataRequest.getParameterGroup("dsParam");
    // 요청 데이터 그룹 내 key 추출
    System.out.println("조직코드: " + dmParam.getValue("strStudNo"));
    System.out.println("조직코드: " + dmParam.getValue("strDeptCd"));

    List<Map<String, Object>> listData = new ArrayList();
    Map<String, Object> row = new HashMap();

    row.put("PARTY_ID", "000000000001");
    row.put("ENR_NM", "Park");
    row.put("STUD_NO", "2009001002");
    row.put("REP_NM", "박길동");
    listData.add(row);
    row = new HashMap();
    row.put("PARTY_ID", "000000000002");
    row.put("ENR_NM", "Kim");
    row.put("STUD_NO", "2009001003");
    row.put("REP_NM", "김대영");
    listData.add(row);

    dataRequest.setResponse("dsCmnTmpReg", listData);

    return new ModelAndView();
}

```

DataRequest로 요청 데이터(dmParam)를 추출하고 DAO 및 mapper실행 후 응답 데이터(dsCmnTmpReg)를 eXBuilder6에게 전달해줍니다.

## 11. 모듈

모듈은 eXBuilder6 프로젝트의 소스 폴더에 포함되어 있으며 \*.modules.js 형태의 파일입니다.

모든 공통 모듈들은 컴파일이 되면 cpr-lib 에 **user-moudels.js** 라는 하나의 파일로 컴파일 됩니다.

**모듈은 반복적인 소스에 대한 비용 절감 및 정형화된 패턴으로 스크립트 표준화하여 개발 생산성에 이바지합니다. 또한 프로젝트 표준에 대한 공통요소 적용을 용이하게 합니다.**

각 공통 모듈들은 고유의 식별자를 가지며, 이를 모듈 URI 또는 모듈 ID 라고 합니다. 모듈의 식별자는 소스폴더로 부터의 상대 경로가 되며, 파일 이름 중 마지막 부분인 .module.js 는 생략 됩니다. 예를 들어, clx-src/mymodules/Test.module.js 의 경우, my-modules/Test 가 모듈의 식별자가 됩니다.

### 11.1. 모듈 출판 방식

기본적으로 공통 모듈 및 앱들은 독자적인 공간을 가지며, 이 공간은 다른 모듈이나 앱들에서 접근이 불가능합니다. 공통 모듈 내부의 변수나 함수를 **외부에서 접근하여 사용하기 위해서는 출판**이라는 과정이 필요합니다. 출판에는 글로벌 출판 및 모듈 멤버 출판 두가지 유형이 존재합니다.

#### 11.1.1. 글로벌 출판

글로벌 출판이란, 명시적인 импорт 과정없이 모든 영역에서 자유롭게 접근 가능한 요소를 출판하는 방법입니다. Globals 키워드를 통해 출판할 수 있습니다. 이후, 프로젝트 전 영역에서 gloMessage 를 사용할 수 있으며 window 객체에 붙어있다고 인식해도 무방합니다.

```

/**
 * 글로벌 메시지입니다.
 * @param {String} psMessage 메시지내용

```

```
*/
globals.gloMessage = function(psMessage){
    alert(psMessage);
}
```

### 11.1.2. 모듈 멤버 출판

모듈 멤버 출판이란, 특정 요소(상수나 함수)를 모듈의 멤버 변수 형태로 출판하는 것을 말합니다. 이렇게 출판한 요소들은 해당 모듈을 사용하는 외부 영역에서, 멤버변수 형태로 접근이 가능합니다.

```
/**
 * 메시지 알람입니다.
 * @param {String} psMessage 메시지내용
 */
function alertMessage(psMessage){
    alert(psMessage);
}
exports.expMessage = alertMessage; //외부에서 expMessage를 통해서 접근 가능합니다.
```

모듈을 출판하려면 **exports** 키워드를 이용하여 외부에서 접근 가능한 모듈을 추가합니다. 모듈 멤버의 이름이 중복되지 않는 한 여러 요소를 출판할 수 있습니다. 이렇게 출판한 expMessage 는 외부(앱)에서 Modules.require 함수를 통해서 임포트 한 뒤 사용할 수 있습니다.

```
/*모듈을 임포트합니다.*/
var moduleVar = cpr.core.Module.require("dev/module/msg");
moduleVar.expMessage("토마토시스템");
```

## 11.2. 세만틱 콘텐츠 어시스트

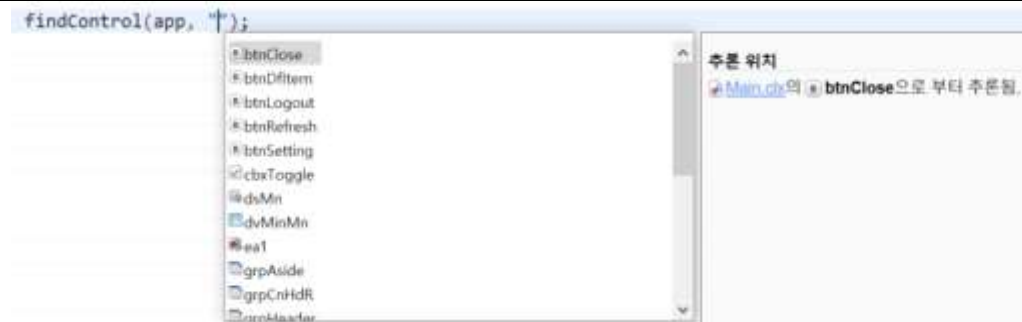
eXBuilder6 를 사용하여 프로젝트의 생산성을 높이기 위해서는 **컨텐츠 어시스트**를 이해하는 것이 매우 중요합니다. 세만틱 콘텐츠 어시스트란 어떤 API 인자가 문자열인 경우, 해당 문자열을 추론하여 제공하는 기능입니다. Param 에 들어갈 정보를 #을 통해서 등록하게 되면 스크립트를 작성할 때 콘텐츠 어시스트를 받을 수 있습니다.

※ 도움말에 eXBuilder6 > 고급 기능 > 공통화 및 재사용 > 주석 작성 규칙(jsdoc)을 참고하시기 바랍니다.

Semantic String 타입	설명
#app	App ID 에 대한 콘텐츠 어시스트 및 링크, 호버 정보가 제공됩니다.
#expression	표현식 문자열을 쉽게 작성할 수 있는 콘텐츠 어시스트 및 링크, 호버 정보가 제공됩니다.

<b>#color</b>	HTML 색상 표기식을 쉽게 입력할 수 있는 콘텐츠 어시스트 및 링크, 호버 정보가 제공됩니다.
<b>#lang</b>	다국어 메시지 키를 쉽게 입력할 수 있는 콘텐츠 어시스트 및 링크, 호버 정보가 제공됩니다.
<b>#control</b>	컨트롤 ID 를 쉽게 입력할 수 있는 콘텐츠 어시스트 및 링크, 호버 정보가 제공됩니다.
<b>#uicontrol</b>	UI 컨트롤 ID 를 쉽게 입력할 수 있는 콘텐츠 어시스트 및 링크, 호버 정보가 제공됩니다.
<b>#컨트롤타입명(소문자)</b>	특정 타입의 컨트롤 ID 를 쉽게 입력할 수 있는 콘텐츠 어시스트 및 링크, 호버 정보가 제공됩니다.

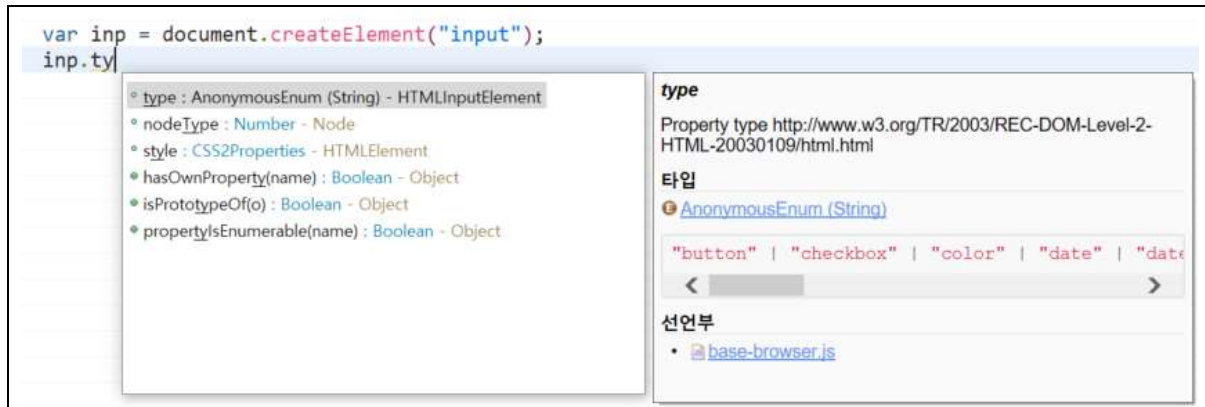
```
/**
 * @param {cpr.core.AppInstance} app 앱인스턴스
 * @param {#control} id 컨트롤ID
 */
globals.findControl = function(app, id){
  ...
}
```



### 11.2.1. 세만틱스 타입 추론

세만틱스 타입 추론이란, 함수의 호출 방법 인자 등을 조합하여 타입을 추론하는 기법입니다.

아래의 예시는 "input" 태그 명을 이용했기 때문에 `inp`의 타입이 `HTMLInputElement`로 추론되어, `type`이라는 속성이 콘텐츠 어시스트 되는 예시를 보여줍니다.



아래의 예시는 이벤트의 속성들이 코드들의 상관 관계를 바탕으로 추론되는 것을 보여줍니다.



### 11.3. 모듈 의존성

공통 모듈은 서로 의존성을 가지고 있습니다. 모듈이 선언되는 동안 다른 모듈에 의존성을 갖는 경우를 **정적인 의존성**이라고 하며, 모듈 특정 기능이 호출되어 수행되는 동안 다른 모듈의 기능을 사용하는 것이 **런타임 의존성**입니다. 의존성 문제로부터 자유로운 방법은 모듈 의존성을 설정하는 방법입니다.

#### 11.3.1. 런타임 의존성

의존성 문제로부터 가장 자유로운 접근법은 정적인 의존성을 제거하는 것입니다. 모듈의 특정 기능이 실제로 수행될 때까지 다른 모듈을 임포트하는 코드를 유예하시기 바랍니다.

##### 권장되지 않는 모듈 작성 패턴

// 모듈이 선언되는 동안 다른 모듈을 로드하여, 정적인 의존성이 발생합니다. 이 시점에 해당 모듈이 선언되어 있지 않을 수도 있기 때문에, 안전하지 않습니다.

```
var otherModule = cpr.core.Module.require("1-Basics/ModuleB");
```

```
exports.test = function(){
    return otherModule.add(1, 5);
};
```

##### 권장 패턴

```
exports.test = function(){
    // 다른 사용자가 test 함수를 사용하는 시점에 모듈을 가져 옵니다.
    var otherModule = cpr.core.Module.require("1-Basics/ModuleB");
    return otherModule.add(1, 5);
};
```

### 11.3.2. 정적 의존성

불가피하게 정적 의존성이 필요한 경우, **module.depends()**를 이용하여 선언할 수 있습니다. 정적인 의존성이 선언되면, 의존 모듈들이 모두 다 선언될 때까지, 현재 모듈의 선언을 보류하게 됩니다.

```
// 정적인 의존성 선언
module.depends("1-Basics/ModuleB");
var otherModule = cpr.core.Module.require("1-Basics/ModuleB");
exports.test = function(){
    return otherModule.add(1, 5);
};
```

정적 의존성을 사용하는 경우, 모듈간의 상호 정적 의존성이나 사이클릭 문제(ex. A->B->C->A 호출)를 주의해야 합니다. 따라서, 불가피한 경우가 아니라면, **런타임 의존성만을 사용하는 것이 바람직합니다.**

정적 의존성을 선언하는 코드는 반드시 모듈의 시작 부분에 작성되어야 하며, 복수의 인자를 이용하여 다중 의존성을 선언할 수 있습니다. 모든 모듈이 전부 로드 되었으나, 의존성이 만족되지 않거나 사이클릭 등이 발생하는 경우, 해당 모듈은 선언되지 않으며, 콘솔에 에러 메시지가 출력됩니다.

## 11.4. 모듈 작성 가이드 라인

1. 모듈이 자바스크립트 프로토타입 디자인 패턴을 이용해 타입을 선언하는 경우, 타입 선언은 반드시 루트 스코프(스크립트 파일 내에서 가장 최상위 영역)에서 하시기 바랍니다. 이 타입이 묵시적으로 다른 소스 파일에서 참조될 경우, 안전하게 타입 인식이 이뤄지지 않을 수도 있습니다.
2. 모듈이 제공하는 타입이 널리 사용되는 경우, globals 키워드를 통해 출판하시기 바랍니다.
3. 이벤트 버스에 필터를 추가하는 코드는 반드시 공통 모듈에서만 작성하시기 바랍니다. 정적인 변경을 일으키는 코드는 공통 모듈에 두는 것이 유지보수에 도움이 됩니다.
4. 표현식 엔진에 함수나 상수를 추가하는 코드는 반드시 공통 모듈에서만 작성하시기 바랍니다.
5. 공통 모듈은 보통 주어진 문자열 인자에 기반하여 작동하므로, 타입을 정확히 예측할 수 없는 경우가 많습니다. 이러한 공통 코드의 작성시 생산성을 높이기 위해, 타입 주석을 이용하여, 콘텐츠 어시스트의 도움을 받을 수 있습니다.

## 12. CI 연동

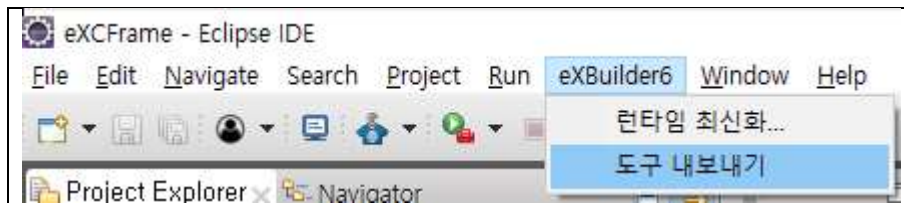
### 12.1. eXBuilder6 도구 내보내기

최신화된 컴파일 버전 또는 프로젝트 버전에 맞는 환경으로 컴파일 내보내기를 하기 위해서 도구 내보내기라는 기능을 사용합니다. eXBuilder6 의 컴파일러는 CI 툴과 연동하여 CI 서버 등에서 eXBuilder6 의 소스를 컴파일 할 수 있도록 지원합니다. eXBuilder6 의 컴파일러를 얻으려면 라이브러리 내보내기 기능을 사용해야 합니다. 라이브러리 내보내기로 내보낼 경우 내보내는 로컬 PC 에 설치된 eXBuilder6 버전으로 컴파일러가 생성됩니다. 일반적으로 CI 환경에서 ant build 를 사용하나 e6-compiler.jar 은 단독으로도 실행할 수 있습니다.

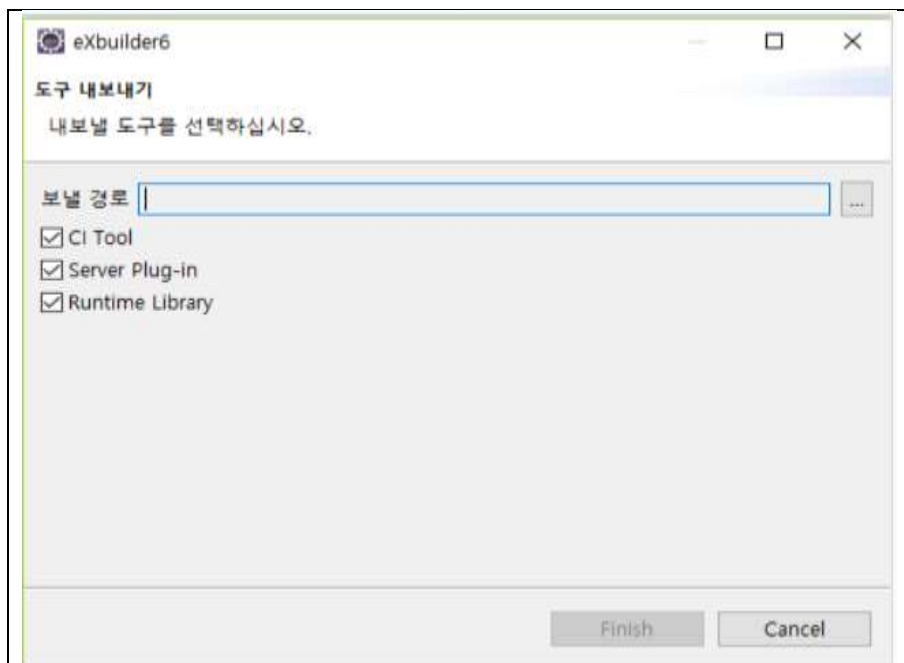
#### ■ 라이브러리 내보내기 방법


라이브러리 내보내기 기능을 사용하여 eXBuilder6 컴파일러를 export 할 수 있습니다.





1) 상단 메뉴에서 eXBuilder6 > 도구 내보내기를 선택합니다.



2) 내보내기 할 경로를 선택하고 [Finish] 버튼을 클릭합니다.



모듈명	설명
<b>CI Tool</b>  e6-compiler.jar	CI 와 연동하거나 단독 수행이 가능한 컴파일러를 내보낼지 여부를 결정합니다.

<b>Server Plug-in</b>  cleopatra_server.jar	eXBuilder6 서버 플러그인 jar 파일을 내보낼지 여부를 결정합니다. (Cleopatra_server.jar) 배포 시 서버 클래스 패스 경로에 위치하도록 설정합니다.
<b>Runtime Library</b>  conf  css  cleopatra.js	eXBuilder6 런타임 파일과 eXBuilder6 런타임 구동에 필요한 Cleopatra.js 와 defaults.js 파일을 내보낼지 여부를 결정합니다. (runtime 폴더) ※ 프로젝트에 설정 된 defaults.js 와 export 된 defaults.js 은 다르니 유념하시길 바랍니다.

## 12.2. 컴파일

e6-compiler.jar 파일의 클래스 패스에 추가하거나 ant build 스크립트에서 아래와 같이 동적으로 클래스 패스를 추가할 수 있습니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="eXCFRAME" default="deploy" basedir=". /">
  <property name="build-lib.dir" location="ci-lib/clx"></property>
  <property name="workspace.dir" location=".. /eXCFRAME-ui"></property>
  <property name="target.dir" location="."></property>

  <taskdef name="clxcompile"
    classname="kr.co.tomatosystem.exbuilder.cikit.build.ant.Compile"
    classpath="${build-lib.dir} /e6-compiler.jar"
  />
```

위의 build-lib.jar 로 매핑된 build-lib 는 도구 내보내기로 내보냈을 시 생성되는 e6-compiler.jar 파일이 위치한 디렉토리입니다.

### 컴파일 예시(clxcompile.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="eXCFRAME" default="deploy" basedir=". /">
  <property name="build-lib.dir" location="ci-lib/clx"></property>
  <property name="workspace.dir" location=".. /eXCFRAME-ui"></property>
  <property name="target.dir" location="."></property>

  <taskdef name="clxcompile"
    classname="kr.co.tomatosystem.exbuilder.cikit.build.ant.Compile"
    classpath="${build-lib.dir} /e6-compiler.jar"
  />

  <target name="deploy">
    <clxcompile
      src="${workspace.dir}"
      out="${target.dir} /target/clx-build"
      minify="true"
      single="false"
      language="true"
      main=""
      relativizelessurl="true"
    >
    <exclude pattern="** /.svn/**" />
    </clxcompile>
  </target>
</project>
```

eXBuilder6 의 컴파일러를 ant task 로 등록하여 clxcompile 이라는 task로 소스디렉토리에 배치된 eXBuilder6 결과물(.clx, .js)을 컴파일합니다.



## 컴파일러 속성 설명

태그/속성	값	설명
src	소스 디렉토리	컴파일할 프로젝트( project이 있는 디렉토리)의 루트 폴더 경로
out	출력 디렉토리	컴파일 결과물이 복제될 디렉토리
minify	true/false	이 옵션을 주면 컴파일 결과물이 압축됩니다
single	true/false	이 옵션이 주어지면, 컴파일 결과물의 단일 js 파일이 됩니다
language	true/false	이 옵션이 주어지면 language.json을 컴파일하여 다국어 지원 스크립트를 함께 생성합니다. eXBuilder6에서 설정한 language.json파일을 다국어 기능을 위해 사용하고 싶은 경우 해당 옵션을 지정하여 language.json을 생성하여 사용할 수 있습니다.
main	시작 앱 URI	이 옵션이 주어지면 index.html이 자동으로 생성됩니다.
exclude	중분 빌드를 제외할 파일	이 옵션은 태그옵션이며 반복사용이 가능합니다. 이 옵션이 주어지면 모든 파일을 다 컴파일 하는 대신, glob 패턴을 이용하여 중분 빌드를 제외할 파일들을 선택할 수 있습니다. 빌드 대상 디렉토리의 콘텐츠를 삭제하지 않고, 중분 컴파일된 파일들만 덮어씁니다. <b>include</b> 옵션과 같이 사용할 경우 우선순위는 <b>exclude</b> 가 먼저 적용됩니다. 해당 옵션은 <b>single</b> (싱글 페이지 애플리케이션)옵션과 함께 사용될 수 없습니다.
include	중분 빌드할 파일	이 옵션은 태그옵션이며 반복사용이 가능합니다. 이 옵션이 주어지면 모든 파일을 다 컴파일 하는 대신, glob 패턴을 이용하여 중분 빌드할 파일들을 선택할 수 있습니다. 빌드 대상 디렉토리의 콘텐츠를 삭제하지 않고, 중분 컴파일된 파일들만 덮어씁니다. 해당 옵션은 <b>single</b> (싱글 페이지 애플리케이션)옵션과 함께 사용될 수 없습니다.
rasterizevectorimages	true/false	이 옵션이 주어지면 SVG파일들을 PNG로 변환합니다
relativizelessurl	true/false	이 옵션이 주어지면 다른 LESS 파일에 임포트된 LESS 파일 내 URL들을 진입 LESS 파일 경로에 맞게 변환합니다.