퍼블리싱 가이드라인

1. 퍼블리싱 포인트

퍼블리싱 포인트란 eXBuilder6 런타임에서 디자이너 및 퍼블리셔가 **개인화 할 수 있도록 의도된 스타일링 지점**들을 의미합니다. **eXBuilder6**에서는 이러한 스타일링이 의도된 단위들을 **스타일러**라고 부르며, 스크립트 및 CSS를 통해 접근하고 수정할 수 있습니다.

1.1. 스타일러의 유형

퍼블리싱 포인트들은 스크립트로는 **스타일러** 객체 형태로, 퍼블리셔 측면으로는 CSS 스타일러 **클래스 셀렉터** 형태로 공급되며, 크게 두 가지 형태로 공급됩니다:

스타일러 셀렉터 유형	설명	예시
컨트롤 타입 클래스 셀렉터 (기본 스타일러 셀렉터)	컨트롤의 최외곽 DOM 요소를 스타일링 하기 위해 공급됩니다. `cl-컨트롤 타입명` 형태로 제공됩니다. 이하 **기본 스타일러**로 지칭합니다.	cl-radiobutton
컨트롤 내부 요소 타입 클래스 셀렉터 (서브 스타일러 셀렉터)	컨트롤을 구성하는 내부 DOM 요소를 스타일링 하기 위해 공급됩니다. `cl-컨트롤 타입명-스타일러 이름` 형태로 공급됩니다. 이하 **서브 스타일러**로 지칭합니다.	cl-radiobutton-item

1.2. 기본 스타일러

기본 스타일러 및 **기본 스타일러 클래스 셀렉터**는 컨트롤 자체의 스타일 요소를 퍼블리싱 하기 위해 공급되며 아래와 같은 형태로 제공됩니다:

`cl-컨트롤 타입명`

예를 들어 **버튼**의 경우 `cl-button`, **라디오 버튼**의 경우 `cl-radiobutton` 클래스 셀렉터를 통하여 퍼블리싱 할 수 있습니다.

1.3. 서브 스타일러

일부 컨트롤은 내부 영역별로 컨트롤 자체와는 **별도의 스타일링 규칙**을 가져야 할 필요를 가진 **시각적 요소**들이 존재합니다. 예를 들어 **라디오 버튼**은 각 **아이템**들에 대해 별도의 스타일링이 필요합니다. 이 때, **아이템**처럼, 특정 컨트롤 내부에서 존재하는 스타일링 단위를 **서브 스타일러**라고 합니다. 각 컨트롤들은 시각적 구조의 복잡도에 따라 여러 **서브 스타일러**를 제공할 수도 있으며, **기본 스타일러** 하나만 제공할 수도 있습니다.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

> 기본스타일러: 컨트롤 외곽 자체의 스타일링을 담당하는 스타일러. 루트 스타일러로도 언급됩니다.

기본 스타일러 및 서브 스타일러들에 대한 정보 얻기

각 컨트롤 별로 사용 가능한 기본 스타일러 및 서브 스타일러의 클래스 명칭들은 스튜디오 도움말 > eXBuilder6 > 앱 개발 > 컨트롤 > UI 컨트롤에서 퍼블리싱 하고자 하는 컨트롤을 선택한 뒤, “스타일 구성요소” 항목에서 확인하실 수 있습니다. 다음의 예시는 아코디언의 “스타일 구성요소” 입니다:

자바 스크립트로 스타일러 접근

자바 스크립트로는 컨트롤의 style 속성을 통해 **기본 스타일러**에 접근 할 수 있으며, **서브 스타일러**들은 **기본 스타일러**의 멤버 속성 형태로서 접근 할 수 있습니다:

```
var radioButton = new cpr.controls.RadioButton();

// 기본 스타일러 조작
radioButton.style.css("color", "red");

// item 서브 스타일러 조작
radioButton.style.item.css("color", "red");
```

CSS로 스타일러 접근

서브 스타일러들은 다음과 같은 형식의 클래스 셀렉터를 공급합니다:

`소속 컨트롤 타입 클래스 셀렉터-스타일러 명칭`

예를 들어 라디오 버튼의 아이템은 다음과 같은 클래스 셀렉터가 제공됩니다:

`cl-radiobutton-item`

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

예시(myRadio는 커스텀 클래스 명칭으로 가정):

```
.cl-radiobutton.myRadio {

  // 기본 스타일러
  color: red;
  .cl-radiobutton-item {

    // item 서브 스타일러
    color: red;
  }
}
```

2. eXBuilder6 렌더링의 이해

2.1. eXBuilder6의 렌더링 구조

eXBuilder6의 중요한 임무 중 하나는, 높은 추상성을 갖는 비즈니스 컨트롤을 표현하는 DOM을 개발자의 요청에 따라 효율적으로 합성하고 그 콘텐츠 관리(Manipulate)하는 하는 것입니다. 예를 들어 Grid 컨트롤은 개발자가 제공한 데이터를 바탕으로 많은 셀을 만들고 효율적으로 DOM 콘텐츠를 관리합니다.

또한 eXBuilder6 플랫폼은 컨트롤들 및 데이터의 상태를 감시하며, 변경이 일어날 때마다 동적으로 가장 적절한 시점에 DOM을 재합성 합니다.

따라서, 특정 시점의 DOM의 상태를 관측한 결과를 바탕으로 퍼블리싱을 수행하면 안 됩니다. DOM은 애플리케이션이 실행되는 동안 역동적으로 변경될 수 있기 때문입니다.

> “ 단, 쉘 컨트롤 및 HTMLSnippet등, 개발자가 직접 DOM 콘텐츠를 통제하는 경우, DOM 형상을 대상으로 하는 퍼블리싱을 수행 할 수 있습니다.

eXBuilder6의 런타임이 업데이트 되면서, 다음의 이유에 따라 컨트롤의 DOM구조는 변경될 수 있습니다:

- 성능 향상
- 기능 추가
- 접근성 향상

2.2. 논리적 스타일링 구조

이러한 환경 내에서 안전한 퍼블리싱이 이루어지려면 DOM을 대상으로 하는 셀렉터를 작성하는 대신, **eXBuilder6**가 제공하는 **컨트롤** 및 **컨트롤의 논리적 구성요소**들을 대상으로 하는 셀렉터를 사용하여야 하며,

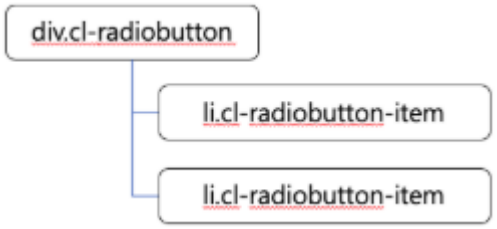
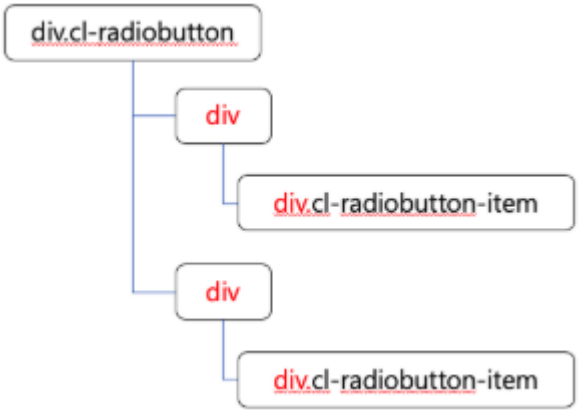
Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

이들을 **기본 스타일러** 및 **서브 스타일러** 셀렉터라고 부릅니다.

각 **기본 스타일러** 및 **서브 스타일러** 사이의 관계 및 구조는 **애플리케이션이 수행되는 동안**이나 혹은 **eXBuilder**의 **버전 업데이트** 등이 이뤄지더라도 변경되지 않습니다. 이렇게 **항상성**이 보장되는 **스타일링 단위** 사이의 **관계 구조**를 **논리적 스타일링 구조**라고 합니다.

> 항상성: 고객의 요청에 기인한 기능 추가 등에 의해 기존 스타일링에 영향을 주지 않는 논리적 스타일링 구조의 추가가 존재할 수는 있습니다.

다음은 API 계약 상 합법적인 라디오 버튼의 DOM 구조 변경의 예시를 나타냅니다:

변경 전	변경 후
	
	item의 태그 명이 li에서 div로 변경되었고, 부모 노드가 추가되었습니다.

위의 예시에서 **라디오 버튼**의 DOM 구조는 변경되었으나, **라디오 버튼** 자체의 스타일은 `.cl-radiobutton`이 결정하며, **아이템**의 스타일은 `.cl-radiobutton-item`이 결정한다는 사실과, 이들이 부모 자식 관계라는 사실에는 변동이 없습니다.

3. 레이아웃 시스템과의 호환성

eXBuilder6는 화면에 표기된 컨트롤들을 렌더링 할 때, 부모의 **레이아웃 시스템**과 자식의 **레이아웃 컨스트레인트** 정보를 조합하여, 그 위치 및 크기를 스크립트로 결정하고, 그 결과물을 표준 HTML/CSS 형태로 반영합니다. 또한 레이아웃 조건 및 컨트롤들의 상태에 따라 결과물의 양상은 크게 변경될 수 있습니다. 예를 들어 TabFolder의 headerPosition 값에 따라, 탭 폴더의 헤더 위치와 연관된 DOM의 순서 등이 변경될 수 있습니다. 또다른 예시로는 개발자가 동적으로 레이아웃 시스템을 변경한 경우, 전체 DOM의 구조가 변경될 수도 있습니다.

> 레이아웃 및 컨스트레인트: 모든 레이아웃은 해당 체계 내에서 자식을 배치하는 방법을 설명하는 컨스트레인트 모델을 갖습니다. 예를 들어 폼 레이아웃의 경우 컨트롤을 배치

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

> 할 행 및 열의 정보가 컨스트레인트가 됩니다.

따라서 퍼블리셔는 **eXBuilder6**가 동적으로 결정하는 **레이아웃 시스템**에 영향을 줄 수 있는 레이아웃 속성들에 대해 스타일링을 수행해서는 아니됩니다. 이 규칙을 준수하기 위하여 CSS또는 LESS파일은 다음의 속성을 지정하는 구문을 포함해서는 아니됩니다.

> “ 레이아웃 속성들: 레이아웃 시스템이 상황에 따라 동적으로 합성하여 만들어내는 CSS 속성들.

속성	이유
display	DOM의 레이아웃 규칙 및 가시성에 영향을 미칩니다. 예를 들어 display를 none으로 강제로 지정한 경우, control.visible = true로 프로그래밍 하여도 화면에 표시되지 않게 됩니다.
position	DOM의 레이아웃 규칙에 영향을 미쳐, 내부 좌표계 전체에 왜곡을 가져오게 됩니다. 모든 컨트롤들은 부모의 레이아웃 시스템에 따라 적절한 position 속성을 부여 받게 됩니다.
left	DOM의 위치및 크기는 eXBuilder6의 레이아웃 시스템 및 개발자가 입력한 레이아웃 컨스트레인트에 의해 결정됩니다.
top	DOM의 위치및 크기는 eXBuilder6의 레이아웃 시스템 및 개발자가 입력한 레이아웃 컨스트레인트에 의해 결정됩니다.
right	DOM의 위치및 크기는 eXBuilder6의 레이아웃 시스템 및 개발자가 입력한 레이아웃 컨스트레인트에 의해 결정됩니다.
bottom	DOM의 위치및 크기는 eXBuilder6의 레이아웃 시스템 및 개발자가 입력한 레이아웃 컨스트레인트에 의해 결정됩니다.
width, min-width	DOM의 위치및 크기는 eXBuilder6의 레이아웃 시스템 및 개발자가 입력한 레이아웃 컨스트레인트에 의해 결정됩니다.
height, min-height	DOM의 위치및 크기는 eXBuilder6의 레이아웃 시스템 및 개발자가 입력한 레이아웃 컨스트레인트에 의해 결정됩니다.

4. UDC의 활용

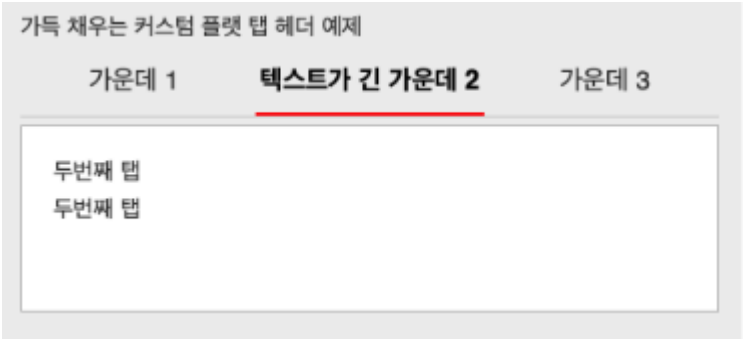
만약 필요로 하는 레이아웃이나 동작을 갖춘 컨트롤이 없는 경우, 안전한 퍼블리싱 구조를 갖는 UDC 컨트롤을 작성할 수 있습니다. UDC 컨트롤 작성을 통해 원하는 형태대로 레이아웃을 중첩하고, 직접 스타일 클래스 명칭을 추가하여, 항상성을 보장받는 논리적 스타일링 구조를 확보할 수 있습니다.

스튜디오의 **eXBuilder 예제 마법사**에서, `“presentation-example”`를 선택하고 예제 프로젝트를 만들어 일부 참고할 만한 예제들을 확인할 수 있습니다:

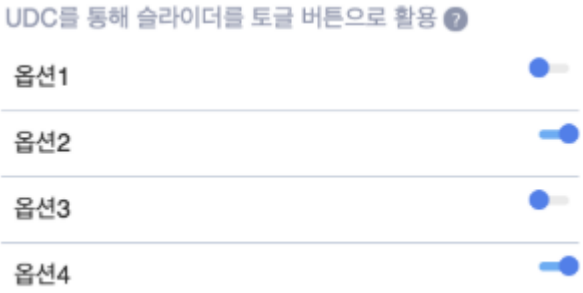
Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

- examples/custom-radio/CustomRadio.clx
  우측에 추가 버튼을 갖는 라디오 버튼 예시

- examples/custom-tabfolder-header/CustomTabHeaderExample.clx
  탭 폴더 헤더를 별도로 만들어 TabFolder와 연동하는 예시



- examples/custom-toggle-button/CustomToggle.clx
  슬라이더를 이용한 토글 버튼 예시



5. 스타일링 지침

안전하게 **지속 가능한 퍼블리싱**을 위하여, 퍼블리싱은 다음 지침을 준수하여야 합니다.

> “ 지속 가능한 퍼블리싱: 런타임을 업데이트 하더라도 항상성을 보장받는 안전한 퍼블리싱

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

번호	규칙	이유
1	CSS의 셀렉터 콤비네이션은 **기본 스타일러**, **서브 스타일러**의 클래스 명 및 **사용자 지정 클래스 명**의 조합으로 이루어져야 합니다.	비 API 요소를 통한 셀렉터는 버전 호환성이 보증되지 않습니다.
2	차일드 셀렉터 컴비네이터(>)를 사용해서는 안됩니다.	2장에서 언급된 것처럼, 런타임 업데이트가 논리적 스타일링의 구조는 변경하지 않으나, DOM의 직계 자손 구조 및 태그 명 등은 변경될 수 있습니다.
3	레이아웃 시스템 및 컨트롤 내부 레이아웃에 영향을 주는 속성을 지정하지 마십시오.	3장에서 언급된 것처럼, 레이아웃과 관련한 CSS 속성들은 런타임이 자동으로 만들어 내므로, 이들과 충돌을 일으킬 수 있습니다.
4	DOM을 관측한 결과를 바탕으로 퍼블리싱을 수행하지 마십시오.	DOM은 애플리케이션이 수행되는 동안 계속해서 바뀌게 됩니다. 예를 들어 컨트롤의 특정 속성이 변경되면, DOM구조 전체가 완전히 변경될 수도 있습니다. 따라서, 반드시 논리적인 스타일링 구조에 따라 퍼블리싱을 수행하여 주십시오.
5	eXBuilder6가 제공하는 컨트롤이 원하는 스타일링 요소를 통제하는 속성을 갖고 있다면, 그것을 CSS로 지정하지 마십시오.	예를 들어 라디오 버튼의 경우 각 아이템 사이의 간격을 결정하는 itemSpacing 속성을 제공합니다. 라디오 버튼이 배치되는 규칙이나 부모의 상황에 따라, 이 값을 DOM에 반영하는 스타일 형태는 크게 변할 수 있습니다. 따라서 특정한 한 유형의 상태에만 의존하여, 컨트롤이 제공하는 기능을 우회하여 CSS로 구현하여서는 안됩니다. 스타일 요소처럼 보이지만, 컨트롤이 관련 속성을 제공하는 경우는, 다양한 문맥에서 정상적인 동작을 보증하기 위하여 불가피한 경우에만 제공됩니다.
6	레이아웃 시스템이 여백과 정렬과 관련한 속성들을 제공하는 경우, 해당 속성을 이용하고 CSS로 지정하지 마십시오.	레이아웃 시스템은 설정에 따라, 동적으로 하위 노드 구조에 많은 변화가 있을 수 있으므로, 이를 CSS로 안전하게 통제하기 어렵습니다.

6. 예외 사항

6.1. 아이콘 스타일러

각 컨트롤들이 제공하는 다수의 **서브 스타일러들** 중 그 이름이 “icon” 혹은 “expander” 등으로 끝나는 **서브 스타일러**들은 아이콘 또는 이미지를 표시하기 위한 **서브 스타일러** 입니다. 이들 **스타일러**는 이미지 및 아이콘의 크기를 변경하기 위하여, width 및 height 속성의 변경을 별도로 허용합니다.

다음의 예시는 스크립트로 **라디오 버튼**의 아이콘의 크기를 변경하는 방법을 보여줍니다:

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

```
radioButton.style.icon.css({
  "width" : "20px",
  "height" : "20px"
});
```

CSS로 같은 작업을 수행하는 예시입니다(my-radio는 사용자 정의 클래스 명):

```
.cl-radiobutton.my-radio {
  .cl-radiobutton-item {
    .cl-radiobutton-icon {
      width: 20px;
      height: 20px;
    }
  }
}
```

6.2. 커스텀 스크롤 바

커스텀 스크롤 바를 사용하거나 사용할 예정인 경우, 그룹 컨트롤의 `padding` 속성을 사용하지 마십시오. 커스텀 스크롤바는 그룹의 `padding`을 이용하여 콘텐트 영역을 축소 시키고 스크롤바를 표시합니다.

7. 유틸리티 슈도 클래스

eXBuilder6는 논리적인 첫번째 행과 마지막 행, 그리고 첫번째 열과 마지막 열에 대한 클래스 셀렉터를 공급합니다.

셀렉터	설명	적용 대상
.cl-first-row	특정 레이아웃 내 첫번째 행에 배치된 자식 컨트롤이나, 컨트롤 내의 첫번째 행에 배치된 아이템에 적용됩니다.	• 라디오 버튼의 첫번째 행에 배치된 아이템 • 체크 박스 그룹의 첫번째 행에 배치된 아이템 • 버티컬 레이아웃이 적용된 컨테이너에 부착된 첫번째 자식 • 폼 레이아웃이 적용된 컨테이너에 부착된 첫번째 행에 존재하는 자식 컨트롤들 • 폼 레이아웃 내 구분선

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

셀렉터	설명	적용 대상
.cl-last-row	특정 레이아웃 내 마지막 행에 배치된 자식 컨트롤이나, 컨트롤 내의 마지막 행에 배치된 아이템에 적용됩니다.	• 라디오 버튼의 마지막 행에 배치된 아이템 • 체크 박스 그룹의 마지막 행에 배치된 아이템 • 버티컬 레이아웃이 적용된 컨테이너에 부착된 첫번째 자식 • 폼 레이아웃이 적용된 컨테이너에 부착된 첫번째 행에 존재하는 자식 컨트롤들 • 폼 레이아웃 내 구분선
.cl-first-column	특정 레이아웃 내 첫번째 열에 배치된 자식 컨트롤이나, 컨트롤 내의 첫번째 열에 배치된 아이템에 적용됩니다.	• 라디오 버튼의 첫번째 열에 배치된 아이템 • 체크 박스 그룹의 첫번째 열에 배치된 아이템 • 폼 레이아웃이 적용된 컨테이너에 부착된 첫번째 열에 존재하는 자식 컨트롤들 • 폼 레이아웃 내 구분선
.cl-last-row	특정 레이아웃 내 마지막 열에 배치된 자식 컨트롤이나, 컨트롤 내의 마지막 열에 배치된 아이템에 적용됩니다.	• 라디오 버튼의 마지막 열에 배치된 아이템 • 체크 박스 그룹의 마지막 열에 배치된 아이템 • 폼 레이아웃이 적용된 컨테이너에 부착된 마지막 열에 존재하는 자식 컨트롤들 • 폼 레이아웃 내 구분선

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

셀렉터	설명	적용 대상
.cl-odd-row .cl-even-row	폼 레이아웃 및 그리드에서 홀수 행 및 짝수 행에 위치하는 컨트롤에 적용됩니다.	• 그리드의 행 • 폼 레이아웃이 배치된 컨트롤 • 폼 레이아웃 내 구분선
.cl-odd-column .cl-even-column	폼 레이아웃에서 홀수 열 및 짝수 열에 위치하는 컨트롤에 적용됩니다.	• 폼 레이아웃에 배치된 컨트롤 • 폼 레이아웃 내 구분선
.cl-nth-of-row-n	폼 레이아웃 내에서 몇 번째 행에 대한 요소인지를 나타냅니다.	• 폼 레이아웃 내 구분선
.cl-nth-of-column-n	폼 레이아웃 내에서 몇 번째 열에 대한 요소인지를 나타냅니다.	• 폼 레이아웃 내 구분선

8. 권고 사항

8.1. before / after 슈도 클래스 셀렉터 사용 주의

`:before` 및 `:after`와 같은 증강 슈도 셀렉터는 실제로 브라우저를 실행하여 DOM을 구성한 후에만 증강 적용이 가능하기 때문에, eXBuilder6 스튜디오는 이들에 대해 디자인 시점의 사전 렌더링을 제공하지 않습니다. 단순한 콘텐츠 삽입의 경우, 개발자가 큰 혼란을 겪지 않겠지만, 레이아웃 변형등을 목적으로 이러한 증강 슈도 셀렉터를 사용하는 경우, 개발자들은 화면의 디자인 결과물을 예측하는데 어려움을 겪을 수 있습니다.

8.2. 불가피한 레이아웃 수정

불가피하게 `position` 혹은 `display` 및 기타 크기 및 위치 관련 속성을 반드시 변경해야 하는 경우, `!important` 키워드를 이용하여야 합니다. **eXBuilder6**는 개발자나 퍼블리셔의 실수로 중요한 레이아웃 속성이 변형되는 것을 막기 위하여, 관련 속성을 변경하는 API요청을 차단(`자바 스크립트로 스타일러 객체의 css() 메서드를 통해 조작한 경우.`)하며, CSS로 그 값이 덮어씌워지는 것을 방지하기 위하여 인라인 속성으로 렌더링하기 때문입니다.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

> !important: 그 속성을 강제로 덮어 써야 한다면, 그것이 API 보증 계약 외의 행위임을 쉽게 인지하기 하기 위하여.

이러한 스타일링은 현재 버전의 런타임에서 정상적으로 작동하는 것처럼 보이더라도, 디자인 편집기 등에서의 정상적인 렌더링 및 향후 버전의 런타임에서의 정상 작동이 보증되지 않습니다.

8.3. 디자인 편집기와의 불일치

일반적으로 디자인 편집기가 의도한 퍼블리싱 결과물을 보여주지 못한다면, API 표준 계약 범위 이외의 요소를 통한 퍼블리싱이 이루어졌을 가능성이 높습니다. 단순히 편집기가 미리 보여주지 못하는 문제에 그칠 수도 있으나, 그 이상의 문제를 추후에 야기할 수도 있으므로, 반드시 전문가의 확인이 필요합니다.

9. 확장 퍼블리싱 포인트

이 장에서는 표준 퍼블리싱이 제공되지 않아 기획 의도 및 특수한 퍼블리싱 의도를 구현할 수 없는 경우를 대비한 **확장 퍼블리싱 포인트**들을 다룹니다. 확장 퍼블리싱 포인트들은 런타임 작동성 및 **버전 호환성**이 제공되지만 디자인 편집기에서는 적용 되지 않습니다.

> 버전 호환성: 기능 개선등의 목적으로 변경되는 경우, 마이그레이션 가이드라인과 함께 변동이 존재할 수 있습니다.9.1. 레이아웃 시스템 관련

9.1. 레이아웃 시스템 관련

컨테이너(그룹) 내 레이아웃 및 자식 컨트롤들의 레이아웃 특성에 대한 퍼블리싱 방법을 기술합니다.

컨테이너는 일반적으로 다음과 같은 구조를 갖습니다:

- .cl-container – 그룹 컨트롤
  - .cl-layout.cl-scrollbar – 레이아웃 루트
    - .cl-layout-content – 레이아웃 내 컨텐트 팬
      - .cl-layout-wrap – 자식 컨트롤에 레이아웃 특성을 적용하기 위한 래퍼
        - .cl-control – 자식 컨트롤

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

> .cl-scrollbar: 레이아웃이 스크롤 가능하도록 설정된 경우에만 이 클래스가 부여됩니다.
>
> .cl-layout-wrap: 래퍼 노드 없이 레이아웃 특성 적용이 가능한 레이아웃에서 이 노드는 생략됩니다.

붉게 표시된 부분은 레이아웃이 설정된 형태에 따라 존재 유무가 결정됩니다.

레이아웃과 관련된 클래스 셀렉터 목록

셀렉터	설명	적용 레이아웃
.cl-layout, .cl-scroll	특정 컨테이너의 레이아웃 시스템이 사용하는 루트 팬을 얻습니다. 해당 레이아웃이 스크롤을 허용하도록 설정되어 있는 경우, cl-scroll 클래스를 동시에 보유합니다.	모든 레이아웃
.cl-layout-content	레이아웃 시스템 내에서 스크롤 되는 전체 영역을 나타내는 컨텐트 팬을 얻을 수 있습니다.	Vertical, Flow
.cl-layout-wrap	특정 컨트롤이 레이아웃에 포함되기 위해 특수한 노드에 래핑되었다면, 이 셀렉터를 통해 얻을 수 있습니다.	Vertical, Flow
.cl-layout-margin, .cl-layout-margin-top .cl-layout-margin-bottom	특정 레이아웃의 마진 관련 속성으로 공간을 확보하기 위한 노드가 만들어졌다면, 이 셀렉터로 선택할 수 있습니다.	Vertical, Flow
.cl-form-placeholder	폼 레이아웃 사용시, 픽셀 기반의 자동 높이나 너비가 사용된 구획이 있는 경우, 최소 크기를 보장하기 위해 만들어진 노드들을 선택할 수 있습니다.	Form 레이아웃

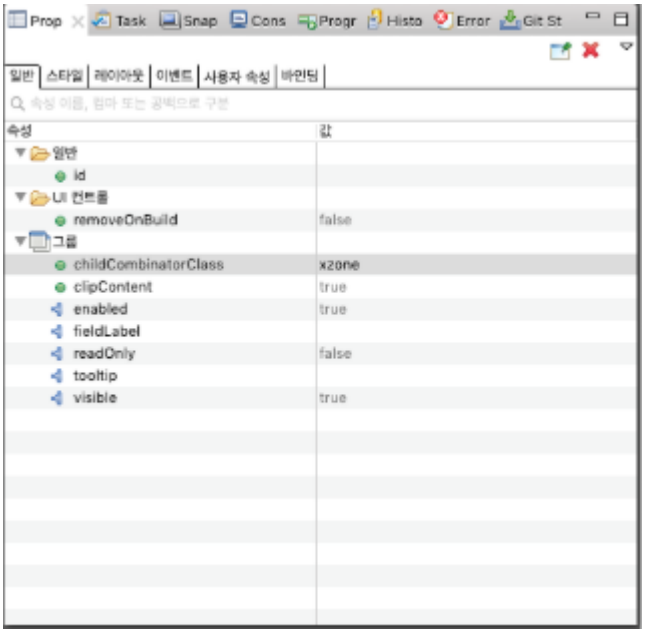
9.2. 차일드 셀렉터 컴비네이터

`9.1`에서 언급된 것 처럼, 레이아웃 마다 자식 컨트롤을 렌더링 하기 위해 만들어 내는 노드의 형태가 각기 다를 수 있습니다. 그러므로 **차일드 셀렉터 컴비네이터**(>)를 사용할 때 주의가 필요합니다.

예를 들어 `.cl-container.group-a > .cl-layout > .cl-control`은 `group-a` 클래스를 가진 컨테이너의 레이아웃이 **XY레이아웃**이거나 **폼 레이아웃**으로 구성 되어 있을 때에는 모든 자식 컨트롤의 노드를 선택하게 되지만, **레이아웃 컨텐트 노드**(`.cl-layout-content`)나 **레이아웃 래퍼 노드**(`.cl-layout-wrap`)를 필요로 하는 레이아웃(`버티컬 레이아웃이나 플로우 레이아웃 등`)으로 구성되어 있을 때는 자식 컨트롤 노드를 선택할 수 없게 됩니다.

이 문제를 해결하려면, 그룹의 **childCombinatorClass** 속성을 사용하십시오:

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.



차일드 컴비네이터 클래스를 지정하면, **직계 자식 컨트롤** 및 직계 해당 컨트롤을 감싸는 의 **레이아웃 래퍼 노드**의 **직계 부모 노드**에 지정한 클래스 명이 클래스 명으로 추가됩니다. 위의 예시에서는 “xzone” 이라는 값을 주었고, 이제 `.xzone > .cl-control` 셀렉터 룰을 이용하여 부모 그룹의 레이아웃 구성이나 설정과 무관하게 항상 childCombinatorClass 값이 “xzone”인 그룹들의 직계 자식 컨트롤을 선택할 수 있게 됩니다.

차일드 컴비네이터 클래스는 직계 자식 컨트롤을 선택하기 위한 셀렉터 사용하기 의도되었기 때문에, 해당 클래스 이름만을 대상으로 하는 룰 블록을 작성해서는 안됩니다. 예를 들어 아래와 같이 차일드 컴비네티얻 클래스 이름인 "xzone" 자체를 대상으로 하는 룰셋을 작성하는 것은 이 가이드라인에 위배 됩니다.

```
.xzone {
  /* 차일드 컴비네이터 클래스 자체를 대상으로
     룰 블록을 작성하여서는 아니됩니다. */

  color : red;
}

/* 아래와 같이 차일드 컴비네이터 클래스 이름은
반드시 직계 자식을 선택하는
차일드 컴비네이터와 조합하여 사용해야 합니다. */
.xzone > .cl-control {
  color: red;
}
```

9.3. 기타 확장 퍼블리싱 포인트

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

셀렉터	설명
.cl-aside	컨테이너(그룹) 내에서 컨트롤을 플롯시키는 API를 사용했을 때, 플로팅 영역을 담당하는 레이어 노드를 얻을 수 있습니다.
.cl-global-aside	다이얼로그, 콤보박스의 드랍다운 선택 상자등, 전역적으로 플로팅되는 영역을 담당하는 레이어 노드를 얻을 수 있습니다.

10. 제한 보증 및 퍼블리싱 포인트 추가 요청

10.1. 퍼블리싱 포인트 추가 요청

적절한 **퍼블리싱 포인트**(`적합한 서브 스타일러 및 서브 스타일러 셀렉터`) 또는 **컨트롤 개인화 포인트**(`컨트롤의 동작이나 형태를 구체적으로 지시하는 속성`)의 부재로 인해 의도한 퍼블리싱의 구현이 불가능한 경우, 이러한 확장점 추가를 토마토 시스템에 요청할 수 있습니다. 토마토 시스템은 요청된 확장점의 타당성에 따라 **퍼블리싱 포인트**를 추가하거나 개선할 수 있습니다.

10.2. 가이드라인 위반 및 귀책 사유

불가피하게 스타일링 가이드라인을 위반한 경우, 클라이언트는 eXBuilder6를 업데이트 할 때마다 이를 검수하여야 하여야 하며, 이로 인한 문제들은 클라이언트의 귀책사유로 취급됩니다.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.