

---

# SmartEditor Basic 2.0

## 개발자 가이드

---

---

# 저작권

---

Copyright © 2012 NHN Corp. All Rights Reserved.

이 문서는 NHN(주)의 지적 재산이므로 어떠한 경우에도 NHN(주)의 공식적인 허가 없이 이 문서의 일부 또는 전체를 복제, 전송, 배포하거나 변경하여 사용할 수 없습니다.

이 문서는 정보 제공의 목적으로만 제공됩니다. NHN(주)는 이 문서에 수록된 정보의 완전성과 정확성을 검증하기 위해 노력하였으나, 발생할 수 있는 내용상의 오류나 누락에 대해서는 책임지지 않습니다. 따라서 이 문서의 사용이나 사용 결과에 따른 책임은 전적으로 사용자에게 있으며, NHN(주)는 이에 대해 명시적 혹은 묵시적으로 어떠한 보증도 하지 않습니다.

관련 URL 정보를 포함하여 이 문서에서 언급한 특정 소프트웨어 상품이나 제품은 해당 소유자가 속한 현지 및 국내외 관련법을 따르며, 해당 법률을 준수하지 않음으로 인해 발생하는 모든 결과에 대한 책임은 전적으로 사용자 자신에게 있습니다.

NHN(주)는 이 문서의 내용을 예고 없이 변경할 수 있습니다.

---

# 문서 정보

---

## 문서 개요

이 문서는 SmartEditor Basic 2.0을 소개하고, 설치 방법과 기능 추가 및 변경 방법을 설명한다.

## 문의처

이 문서의 내용에 오류가 있거나 내용과 관련한 의문 사항이 있으면 아래 사이트에서 문의한다.

<http://dev.naver.com/projects/smarteditor>

## 문서 버전 및 이력

버전	일자	이력사항
1.3	2012.12.12	2.2.1 버전 업데이트 관련 '툴바 버튼의 위치 변경과 버튼 제거' 추가, '새로운 기능 추가' 수정, '사진 쿼 업로더' 수정
1.2	2012.08.02	2.0 버전 설치 방법 수정
1.1	2012.04.25	사진 쿼 업로더 추가
1.0	2012.03.27	최종 배포

---

---

# 목차

---

<b>소개</b>	<b>9</b>
SmartEditor Basic 2.0 의 특징	9
지원 브라우저	10
제공 기능 상세	10
글꼴	10
글자 크기	11
글자 효과	12
글자 색	13
글자 배경 색	14
위 첨자와 아래 첨자	15
정렬	16
줄 간격	16
번호 매기기	17
글머리 기호	17
내어쓰기와 들여쓰기	17
인용구 스타일	18
하이퍼링크	18
특수 기호	19
표 생성과 간단 편집기	20
찾기와 바꾸기	21
편집 모드	22
<b>설치하기</b>	<b>25</b>
설치 전 확인 사항	25
다운로드	25
에디터 구조	27
2.0 버전 설치	27
0.3.x 버전에서 업그레이드	29
<b>기능 추가 및 변경하기</b>	<b>31</b>

---

기본 클래스 함수 추가 및 변경	31
기본 기능 삭제	32
툴바 버튼의 위치 변경과 버튼 제거	33
새로운 기능 추가	36
UI 추가	36
플러그인 등록	38
플러그인 제작	39
<b>사진 퀵 업로더</b>	<b>42</b>
소개	42
파일 구성	43
설치하기	45
사진 플러그인 추가	45
툴바에 사진 버튼 추가	46
PHP 서버 설정	46
팝업 JavaScript 수정	47
사진 퀵 업로더 사용해 보기	50
<b>부록. jindo.FileUploader</b>	<b>51</b>
jindo.FileUploader의 특징	51
초기화	51
요청 수행 과정	51
커스텀 이벤트	52
메서드	53

---

# 표 및 그림 목록

---

## 표 목록

표 1 지원 브라우저와 운영체제	10
표 2 글자 효과의 단축키	13
표 3 들여쓰기와 내어쓰기의 단축키	17
표 4 하이퍼링크 설정 단축키	18
표 5 특수 기호 종류	19
표 6 찾기과 바꾸기의 단축키	22
표 7 배포 파일의 구성과 설명	26
표 8 제공되는 파일 설명	44
표 9 jindo.FileUploader 구성 요소	47
표 10 callAjaxForHTML5() 함수 구성 요소	49
표 11 setPhotoToEditor 함수 구성 요소	49
표 12 jindo.FileUploader 메서드	53

## 그림 목록

그림 1 툴바 더보기 버튼	10
그림 2 글꼴 선택 레이어	11
그림 3 글꼴 적용 예	11
그림 4 글자 크기 선택 레이어	12
그림 5 글자 크기 적용 예	12
그림 6 글자 효과 적용 예	13
그림 7 글자 색 기본 색상 표	13
그림 8 글자 색 컬러 팔레트	14
그림 9 글자 배경색 의 기본 색상 표	14
그림 10 글자 배경 색의 컬러 팔레트	15
그림 11 글자 색과 글자 배경 색 적용 예	15
그림 12 위 첨자와 아래 첨자 적용 예	15
그림 13 문장 정렬 적용 예	16
그림 14 줄 간격 설정 레이어	16
그림 15 줄 간격 200% 적용 예	16

---

그림 16 번호 매기기 적용 예	17
그림 17 글머리 기호 적용 예	17
그림 18 내어쓰기와 들여쓰기 적용 예	17
그림 19 인용구 스타일 레이어	18
그림 20 인용구 적용 예	18
그림 21 하이퍼링크 설정 레이어	18
그림 22 특수 기호 삽입 레이어	19
그림 23 특수 기호 삽입 예	19
그림 24 표 생성 레이어	20
그림 25 표 스타일 선택 레이어	20
그림 26 표 간단 편집기 레이어	21
그림 27 찾기/바꾸기 레이어	21
그림 28 단어 찾기 예	21
그림 29 단어를 찾아 모두 바꾸기 예	22
그림 30 Editor 모드에서 글을 작성하는 예	23
그림 31 HTML 모드에서 글을 작성하는 예	23
그림 32 TEXT 모드에서 글을 작성하는 예	24
그림 33 배포 파일 구성	26
그림 34 SmartEditor2.html의 구조	27
그림 35 툴바의 버튼 그룹	33
그림 37 툴바 버튼의 위치 변경	35
그림 38 툴바 버튼의 제거	36
그림 39 버튼을 추가한 에디터 툴바 화면	38
그림 40 액티브 레이어를 추가한 에디터 화면	40
그림 41 TimeStamper 실행한 에디터 화면	40
그림 42 HTML5 지원 브라우저의 사진 쿼 업로더	42
그림 43 HTML5 미지원 브라우저의 사진 쿼 업로더	43
그림 44 사진 쿼 업로더 팝업 HTML 및 JavaScript 파일 구성	44
그림 45 사진 쿼 업로더 팝업의 image 구성	45
그림 46 사진 쿼 업로더의 plugin 구성	45
그림 47 사진 쿼 업로더의 PHP 서버 구성	47





# 소개

SmartEditor Basic 2.0은 JavaScript로 구현된 웹 기반의 WYSIWYG 편집기이다. 글꼴, 글자 크기, 줄 간격 등을 자유롭게 설정할 수 있으며, 단어 찾기/바꾸기와 같은 편리한 기능을 제공한다.

이 장에서는 SmartEditor Basic 2.0의 특징과 기능을 소개한다.

## SmartEditor Basic 2.0의 특징

SmartEditor Basic 2.0에서 이전 버전에 비해 개선된 기능과 추가된 기능은 다음과 같다.

### 개선된 기능

- UI 디자인 개선  
글꼴, 글자 크기, 줄 간격 등을 설정하는 UI를 기존의 드롭다운 메뉴에서 레이어로 변경하여 디자인이 개선되었다.
- 글자 색과 글자 배경 색  
기존의 기본 색상표 이외에 다양한 색상을 선택할 수 있는 컬러 팔레트를 제공한다.
- 줄 간격  
줄 간격 값을 직접 입력할 수 있다.
- 인용구  
인용구 스타일이 기존의 7가지에서 10가지로 늘어났다.
- 표  
표를 삽입할 때 기존에는 테두리 색상과 두께만 설정할 수 있었지만 SmartEditor Basic 2.0 버전에서는 테두리 스타일을 설정할 수 있으며, 쉽게 표를 꾸밀 수 있도록 표 스타일을 제공한다.

### 추가된 기능

- 표 간단 편집기  
표 편집 기능을 제공하여 표를 삽입한 후에도 스타일을 편집할 수 있다.
- TEXT 모드  
Editor 모드(WYSIWYG)와 HTML 모드 외에 TEXT 모드를 추가하여 단순 텍스트만으로 간단하게 본문의 내용을 작성할 수 있다.

## 지원 브라우저

일부 브라우저는 SmartEditor Basic 2.0을 지원하지 않거나 일부 기능만을 지원한다. 지원하는 기능은 브라우저 특성에 따라 다를 수 있다. 운영체제별로 SmartEditor Basic 2.0의 모든 기능을 지원하는 브라우저는 다음과 같다.

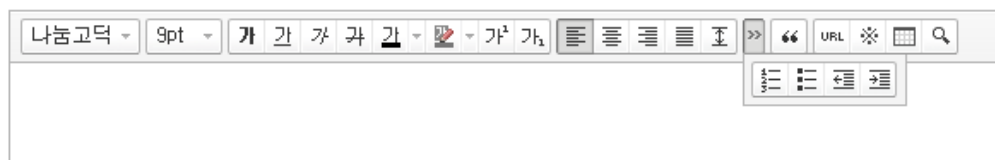
**표 1 지원 브라우저와 운영체제**

운영체제	브라우저
Windows XP	Internet Explorer 7.x, 8.x, Firefox 3.5 이상, Chrome 4.0 이상, Safari 4.0 이상
Windows Vista	Internet Explorer 7.x, 8.x
Windows 7	Internet Explorer 8.x, 9.x, Firefox 3.5 이상, Chrome 4.0 이상, Safari 4.0 이상
Mac	Firefox 3.5 이상, Safari 4.0 이상, Chrome 5.0 이상

SmartEditor Basic 2.0 버전에서는 기존의 SmartEditor Basic 0.3.17 버전에서 제공하던 기능을 개선하고, 새로운 기능을 추가하였다. 또한 Editor 모드(WYSIWYG)와 HTML 모드 외에 TEXT 모드를 추가하여 다양한 방식으로 글을 편집할 수 있다.

## 제공 기능 상세

SmartEditor Basic 2.0이 제공하는 각 기능을 툴바의 아이콘 순서로 알아본다. 번호 매기기 버튼, 글머리 기호 버튼, 내어쓰기 버튼, 들여쓰기 버튼은 툴바의 더보기 버튼(>>)을 클릭하면 나타난다.



**그림 1 툴바 더보기 버튼**

## 글꼴

11가지의 기본 글꼴과 2가지의 나눔글꼴을 제공한다. 제공하는 기본 글꼴은 돋움, 돋움체, 굴림, 굴림체, 바탕, 바탕체, 궁서, Arial, Tahoma, Times New Roman, Verdana이며, 나눔글꼴은 나눔고딕과 나눔명조이다.

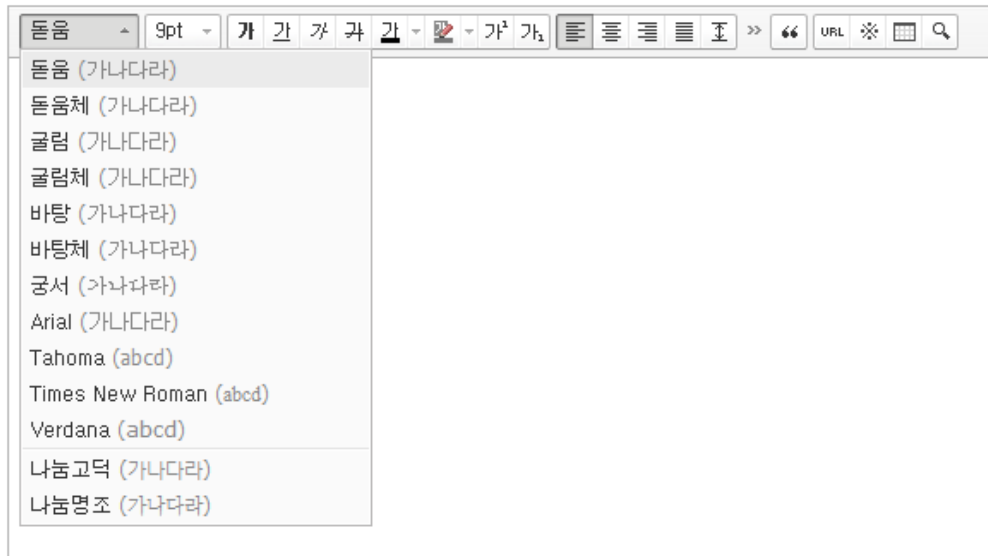


그림 2 글꼴 선택 레이어

나눔고딕과 나눔명조는 브라우저 설치형 웹폰트이며 나눔글꼴

사이트(<http://hangeul.naver.com/font>)에서 내려받아 설치할 수 있다. 나눔글꼴은 서버에 설치하는 폰트가 아니라는 점을 주의한다.

각 글꼴을 적용한 모습은 다음 그림과 같다.

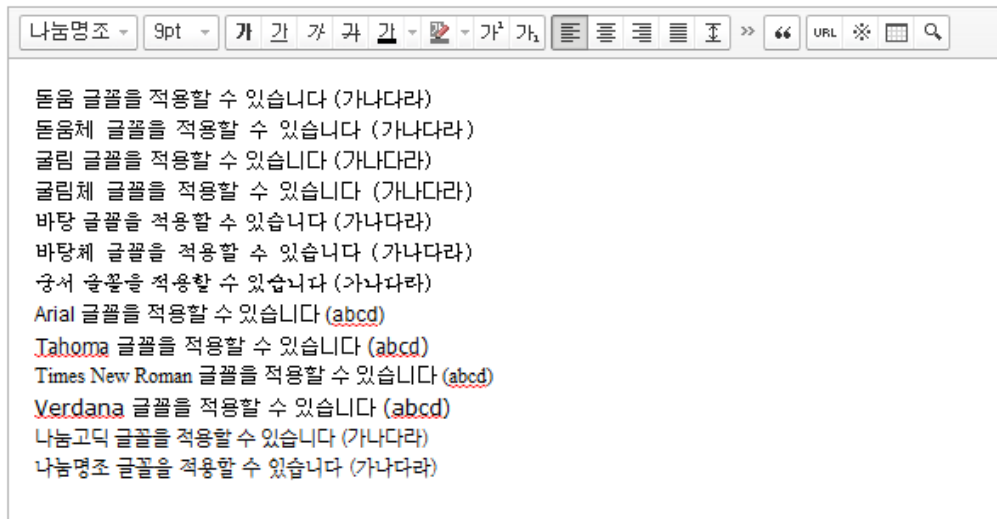


그림 3 글꼴 적용 예

## 글자 크기

글자 크기는 7pt부터 36pt까지 설정할 수 있다.

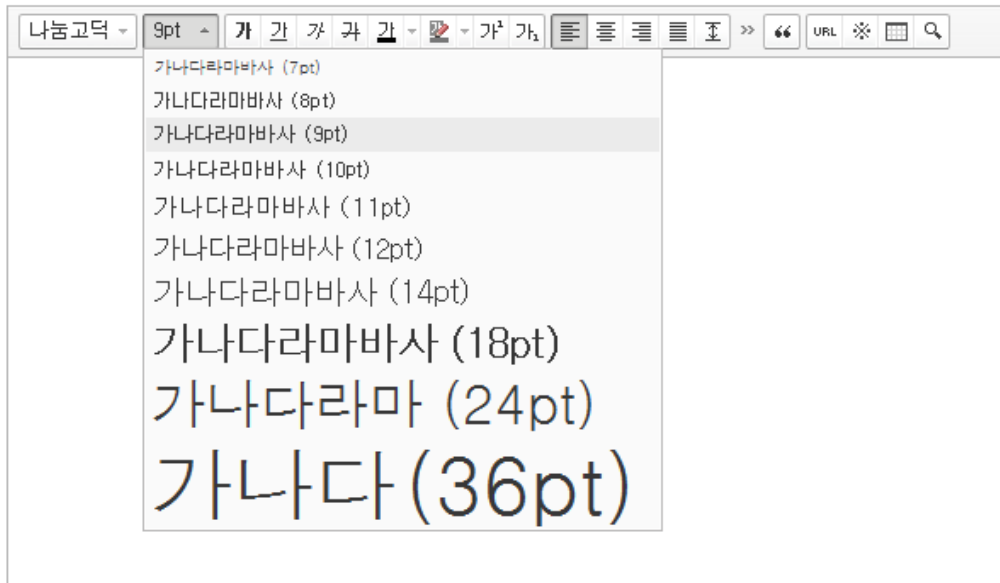


그림 4 글자 크기 선택 레이어

나눔고딕에 각 글자 크기를 적용한 모습은 다음 그림과 같다.

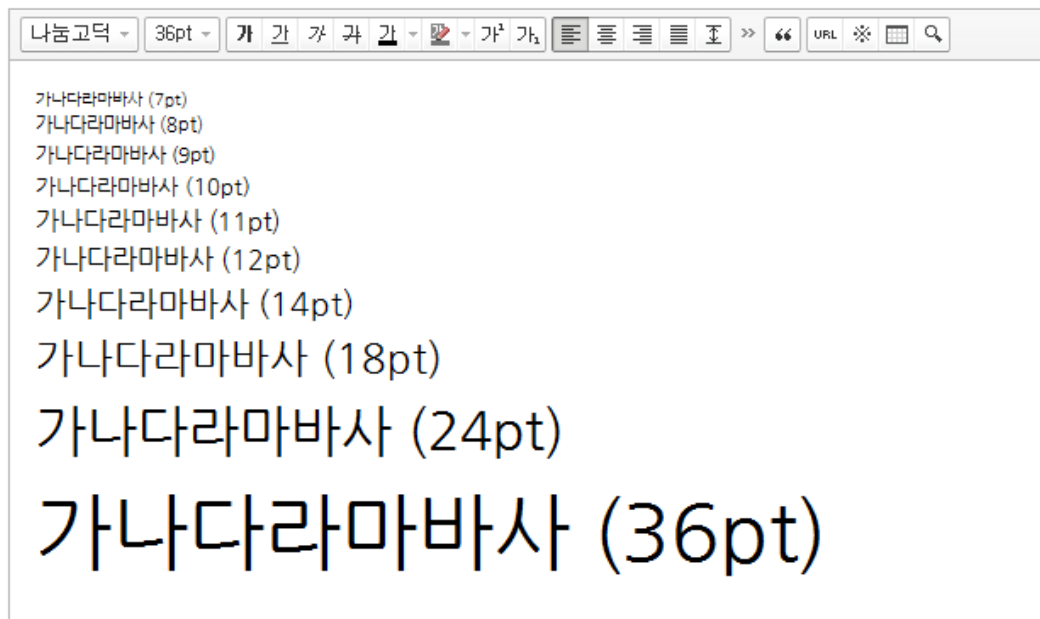


그림 5 글자 크기 적용 예

## 글자 효과

글자에 굵게, 밑줄, 기울임꼴, 취소선 효과를 적용할 수 있다. 각 글자 효과를 적용한 모습은 다음과 같다.

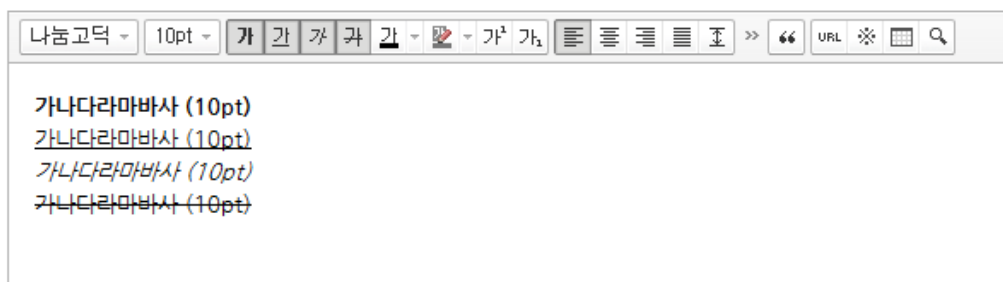


그림 6 글자 효과 적용 예

글자 효과는 단축키를 사용하여 적용할 수도 있다. 각 글자 효과의 단축키는 다음과 같다.

표 2 글자 효과의 단축키

효과	단축키
굵게	Ctrl+B
밑줄	Ctrl+U
기울임	Ctrl+I
취소선	Ctrl+D

## 글자 색

기본 색상 표에서 제공하는 색을 선택하거나 컬러 팔레트에서 색을 선택하여 글자 색을 변경할 수 있다. 기본 색상 표 오른쪽 아래의 **더보기**를 클릭하면 컬러 팔레트가 나타난다. 원하는 색상의 웹 색상 코드 값을 직접 입력할 수도 있다.

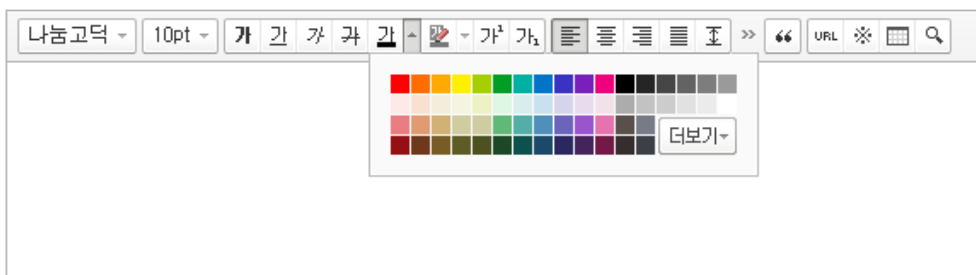


그림 7 글자 색 기본 색상 표

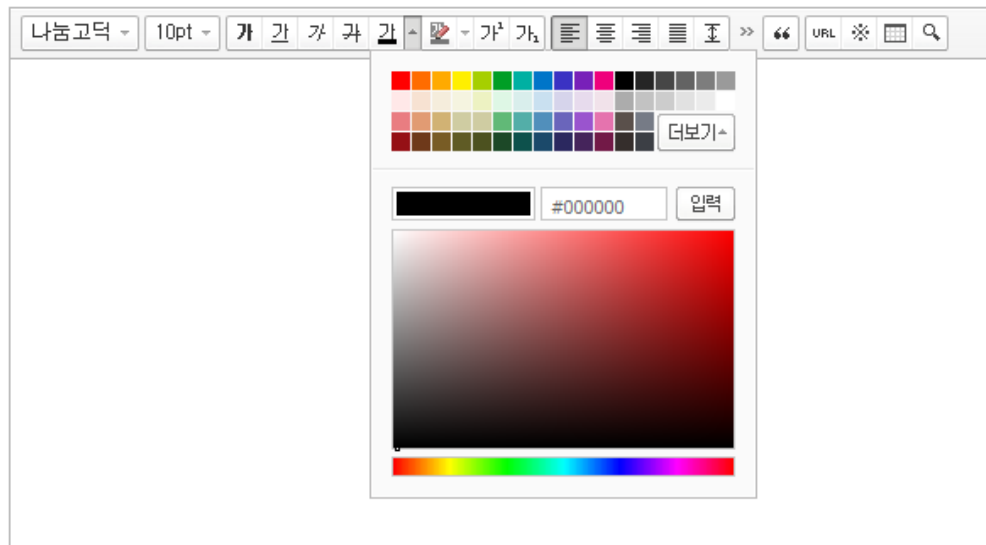


그림 8 글자 색 컬러 팔레트

## 글자 배경 색

기본 색상 표에서 제공하는 색을 선택하거나 컬러 팔레트에서 색을 선택하여 글자 배경 색을 변경할 수 있다. 기본 색상 표 오른쪽 아래의 **더보기**를 클릭하면 컬러 팔레트가 나타난다. 원하는 색상의 웹 색상 코드 값을 직접 입력할 수도 있다.

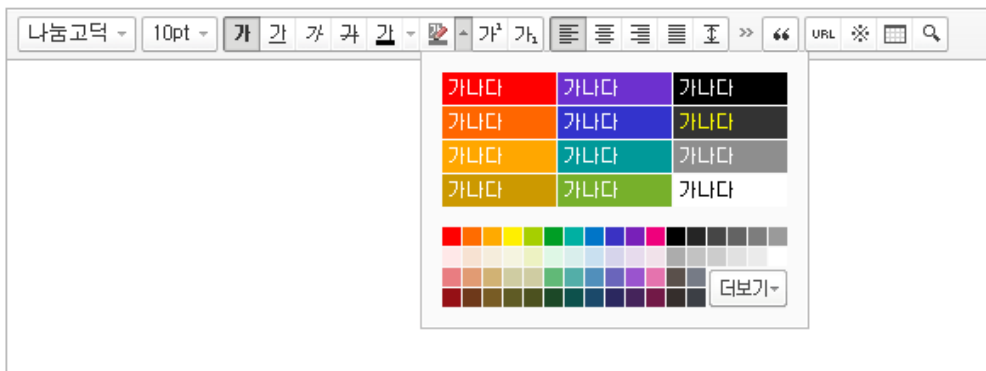


그림 9 글자 배경색 의 기본 색상 표

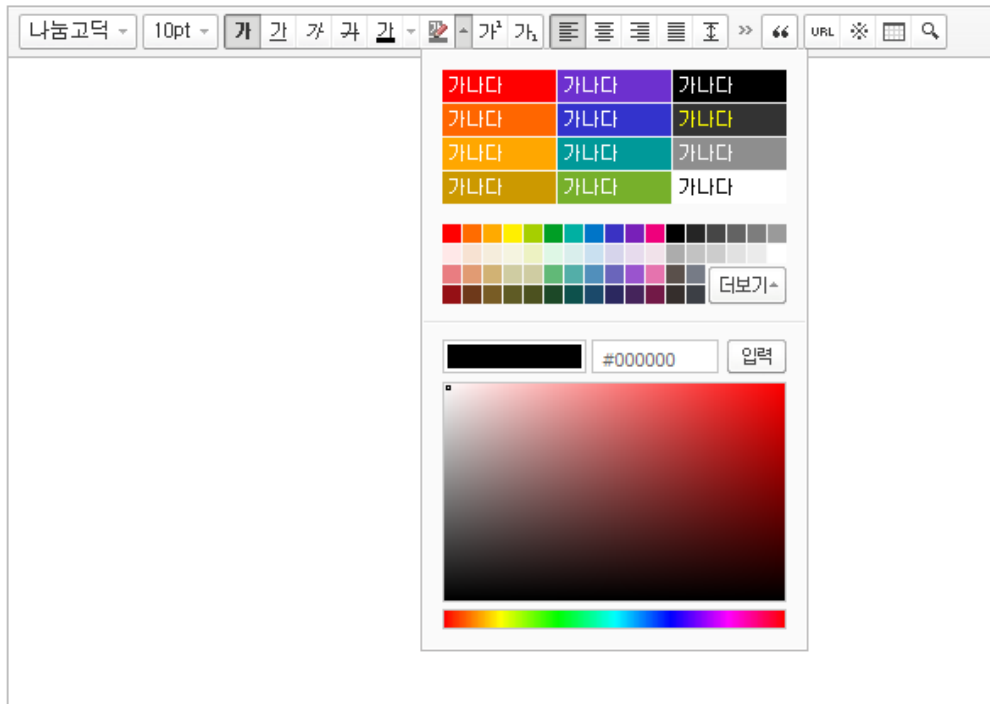


그림 10 글자 배경 색의 컬러 팔레트

글자 색과 글자 배경 색을 적용한 모습은 다음 그림과 같다.

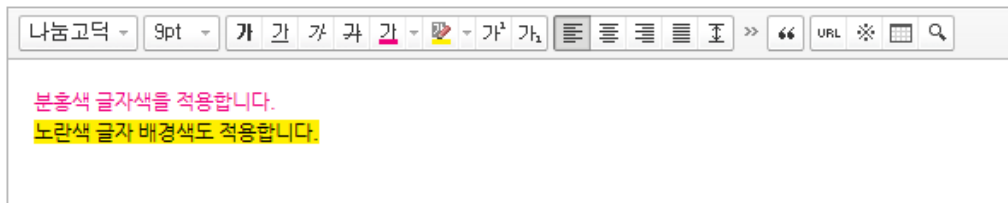


그림 11 글자 색과 글자 배경 색 적용 예

## 위 첨자와 아래 첨자

글자에 위 첨자나 아래 첨자를 적용할 수 있다. 위 첨자와 아래 첨자를 적용한 모습은 다음 그림과 같다.

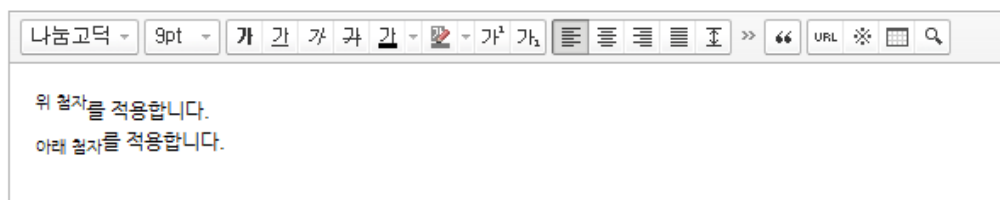


그림 12 위 첨자와 아래 첨자 적용 예

## 정렬

문장을 왼쪽, 가운데, 오른쪽, 양쪽으로 정렬할 수 있다. 문장에 각 정렬 방식을 적용한 모습은 다음 그림과 같다.

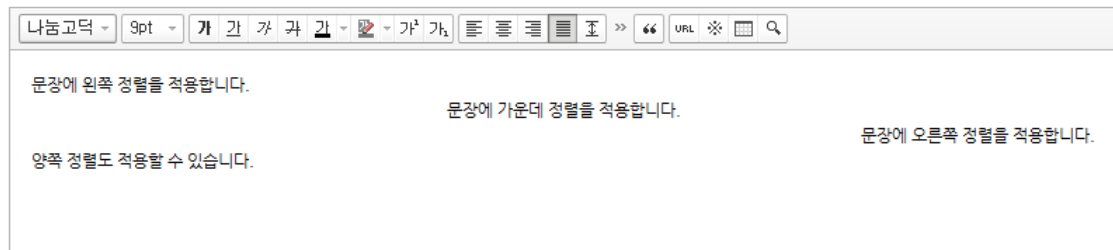


그림 13 문장 정렬 적용 예

## 줄 간격

문장의 줄 간격을 조절할 수 있다. 줄 간격은 50%에서 200%까지의 값 중에서 선택하거나 값을 직접 입력할 수 있다.

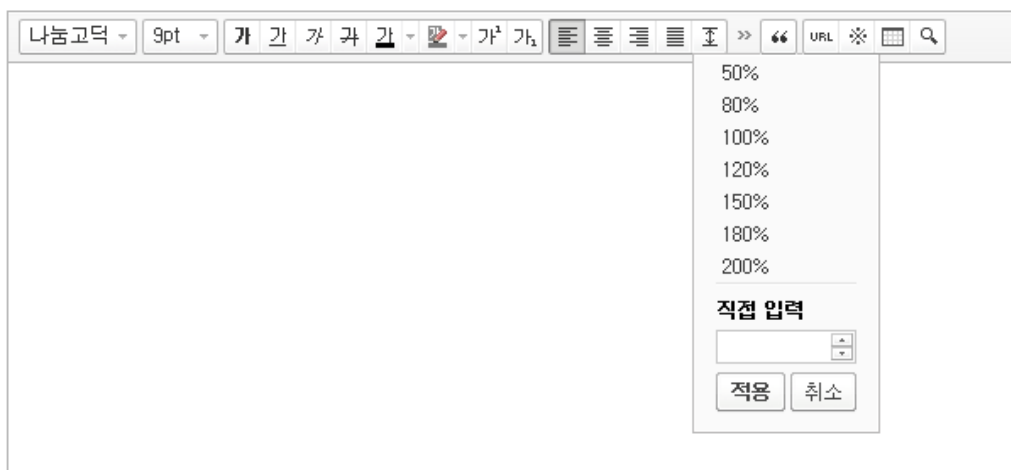


그림 14 줄 간격 설정 레이아웃

줄 간격을 적용한 모습은 다음 그림과 같다.

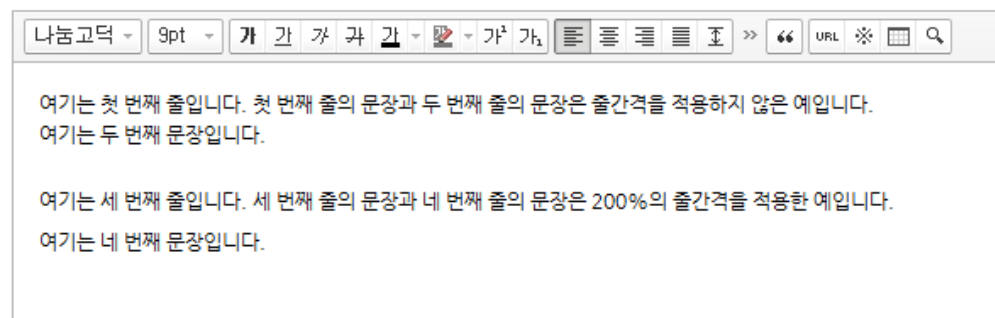


그림 15 줄 간격 200% 적용 예



## 번호 매기기

문장의 앞에 순차적으로 번호를 매길 수 있다. 문장에 번호 매기기를 적용한 모습은 다음 그림과 같다.

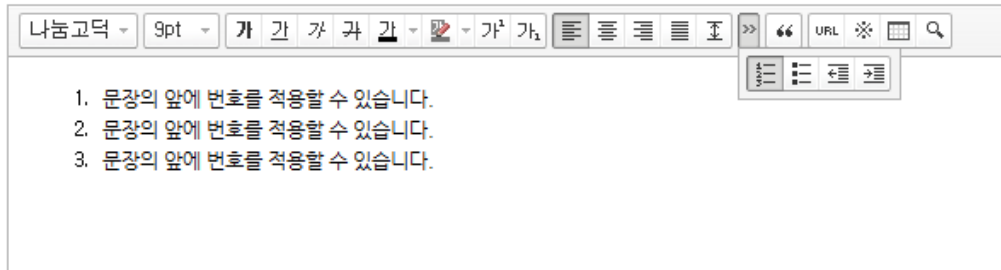


그림 16 번호 매기기 적용 예

## 글머리 기호

문장 앞에 글머리 기호를 붙여 순서 없는 목록을 만들 수 있다. 문장에 글머리 기호를 적용한 모습은 다음 그림과 같다.

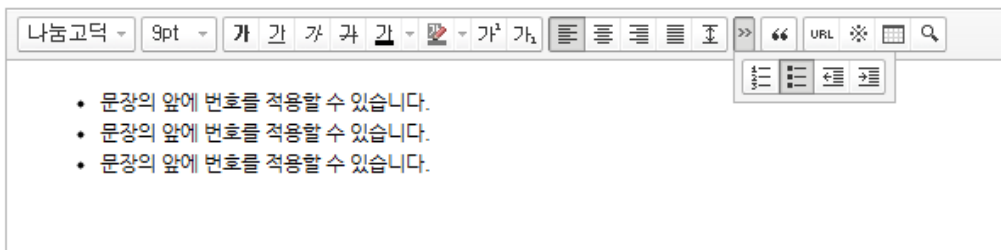


그림 17 글머리 기호 적용 예

## 내어쓰기와 들여쓰기

문장을 내어 쓰거나 들여 쓸 수 있다. 문장에 내어쓰기와 들여쓰기를 적용한 모습은 다음 그림과 같다.

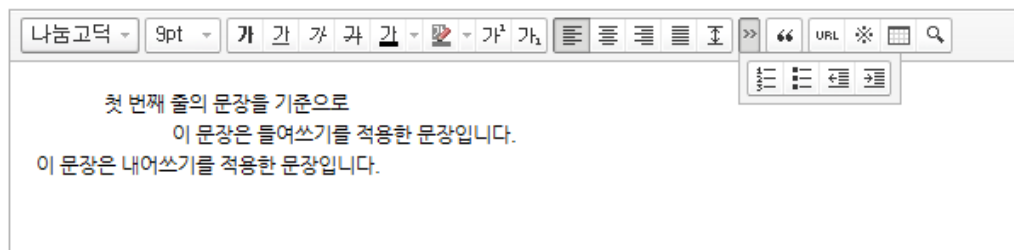


그림 18 내어쓰기와 들여쓰기 적용 예

들여쓰기와 내어쓰기는 단축키를 사용하여 적용할 수도 있다. 각 단축키는 다음과 같다.

표 3 들여쓰기와 내어쓰기의 단축키

종류	단축키
내어쓰기	Shift+Tab

종류	단축키
들여쓰기	Tab

## 인용구 스타일

문장에 인용구 스타일을 적용할 수 있다. 인용구 스타일의 종류는 모두 10가지이다. 인용구 스타일을 적용한 문장을 선택하고 **적용 취소**를 클릭하면 인용구 스타일 적용이 취소된다.

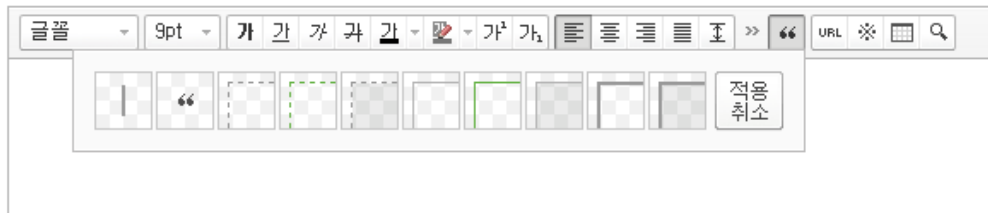


그림 19 인용구 스타일 레이어

문장에 일곱 번째 인용구 스타일을 적용한 모습은 다음 그림과 같다.

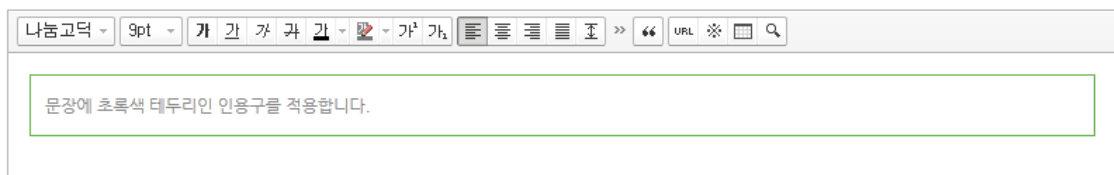


그림 20 인용구 적용 예

## 하이퍼링크

글자에 하이퍼링크를 설정할 수 있다. 하이퍼링크로 연결된 페이지는 새 창에서 열린다.

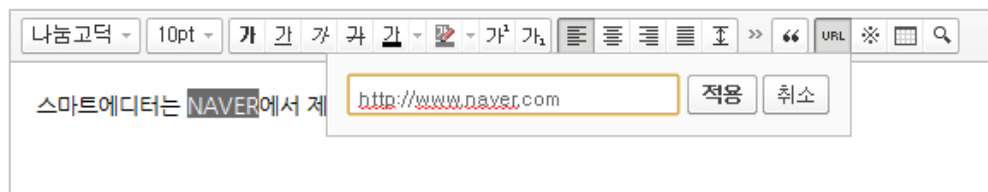


그림 21 하이퍼링크 설정 레이어

하이퍼링크는 단축키를 사용하여 설정할 수도 있다. 단축키는 다음과 같다.

표 4 하이퍼링크 설정 단축키

종류	단축키
링크	Ctrl+K

## 특수 기호

키보드로 입력하기 어려운 특수 기호를 쉽게 삽입할 수 있다.

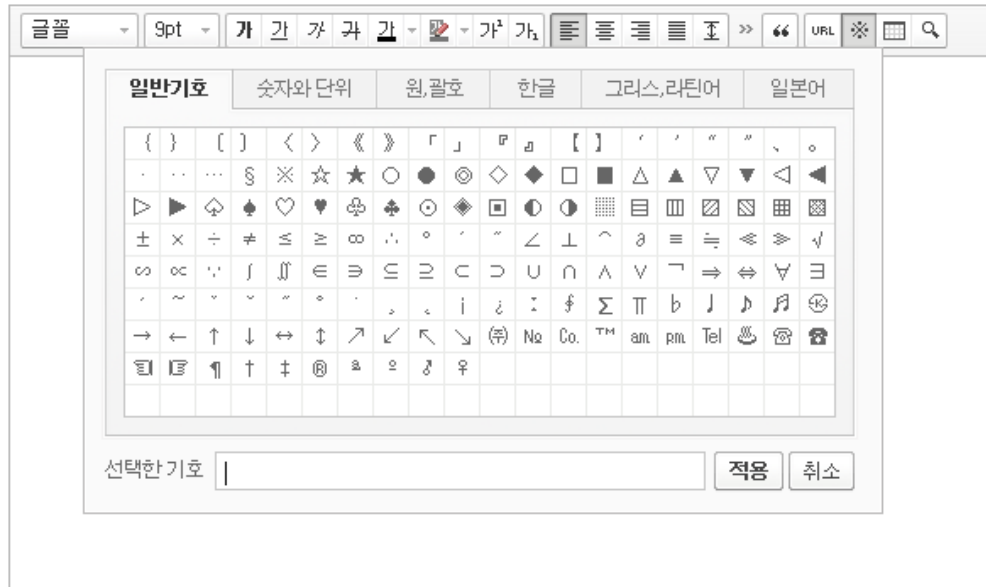


그림 22 특수 기호 삽입 레이어

특수 기호 삽입 레이어의 각 탭은 특수 기호의 종류로, 각각 다음과 같은 문자들을 포함하고 있다.

표 5 특수 기호 종류

특수 기호 종류	예
일반 기호	§ ※ ☆ ★ ▶ 🔍 📄
숫자와 단위	Hz € £ Å °F
원, 괄호	㉠ ㉡ ㉢ ㉣ ㉤ ㉥
한글	ㄱ ㄴ ㄷ ㄹ ㅁ
그리스, 라틴어	Π Ρ Σ π Ø
일본어	あ ざ じ す ね

다음 그림은 문장에 라틴어 특수 기호를 삽입한 예이다.

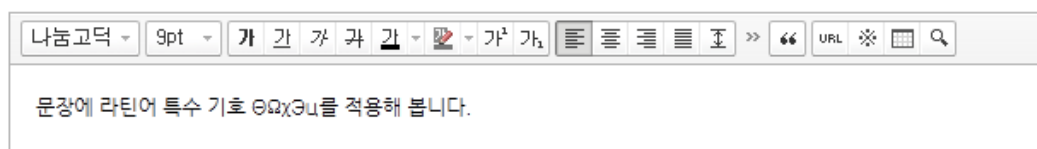


그림 23 특수 기호 삽입 예

## 표 생성과 간단 편집기

SmartEditor Basic 2.0은 표 관련 기능으로 표 생성 기능과 표 편집 기능을 제공한다.

표를 생성할 때에는 행과 열의 개수를 지정할 수 있으며, 테두리 스타일, 테두리 두께, 테두리 색, 셀 배경색을 직접 입력하거나 제공되는 표 스타일 중에서 선택할 수 있다.

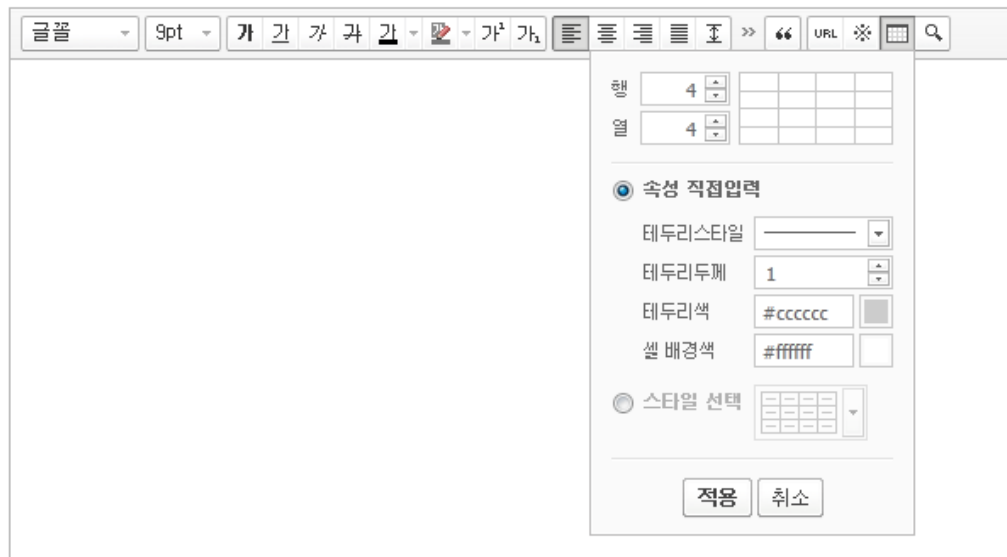


그림 24 표 생성 레이아웃

스타일 선택을 선택하면 다음과 같이 기본으로 제공되는 표 스타일 중에서 선택할 수 있다.

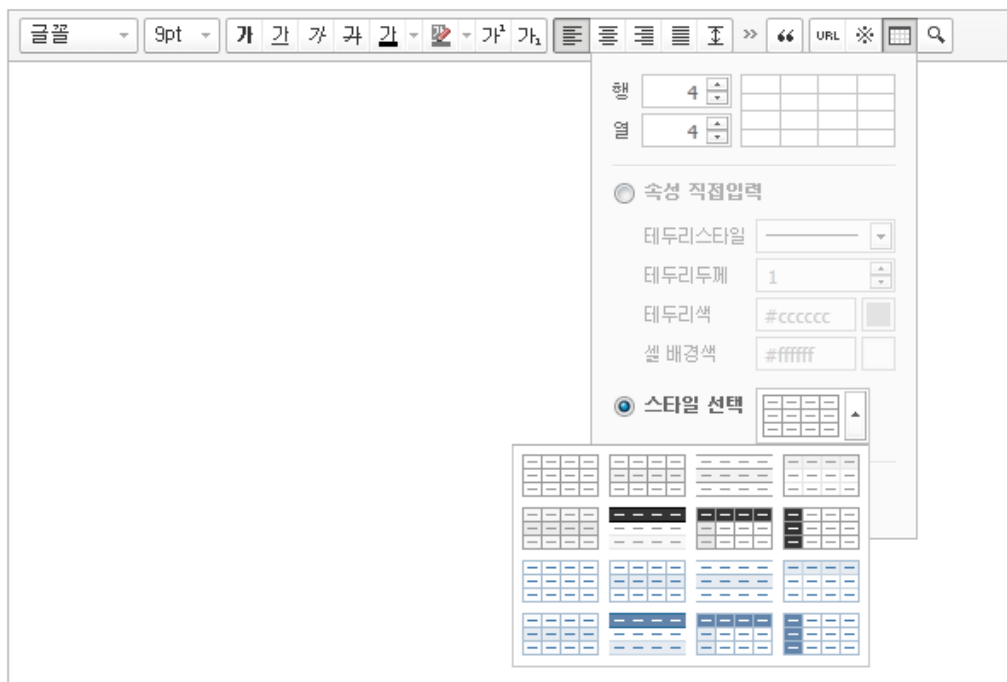


그림 25 표 스타일 선택 레이아웃

표를 생성한 후에도 표의 스타일을 변경할 수 있으며, 셀을 삽입/삭제/분할/병합하거나 특정 셀의 배경색만 변경할 수도 있다. 마우스로 드래그하여 셀을 선택하면 다음과 같이 표 간단 편집기 레이어가 나타난다.

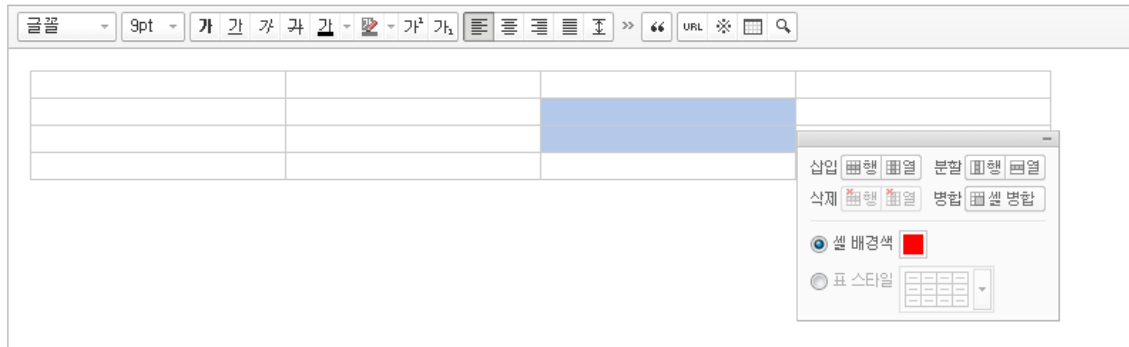


그림 26 표 간단 편집기 레이어

## 찾기와 바꾸기

작성한 글에서 특정 단어 혹은 문장을 찾거나, 찾은 단어나 문장을 다른 단어나 문장으로 바꿀 수 있다.

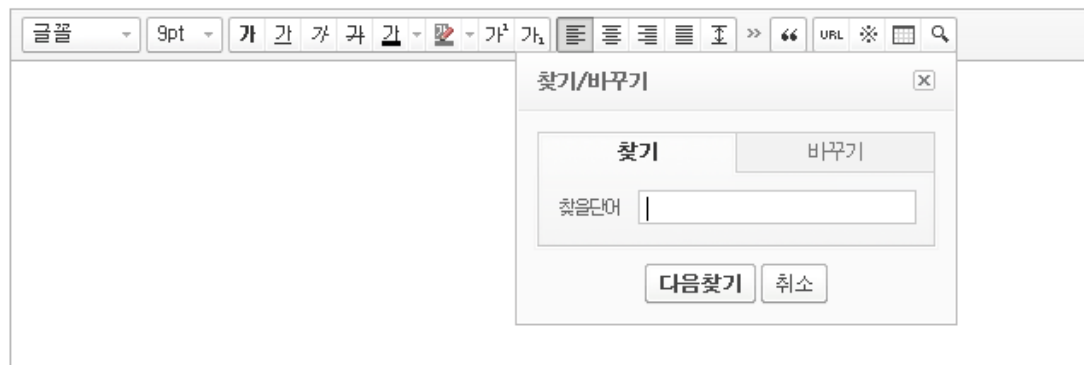


그림 27 찾기/바꾸기 레이어

찾기/바꾸기 레이어는 드래그 앤드 드롭으로 에디터 화면 내에서 위치를 자유롭게 옮길 수 있는 플로팅 레이어이다. 다음 그림은 글에서 '스마트에디터'라는 단어를 찾은 예이다.

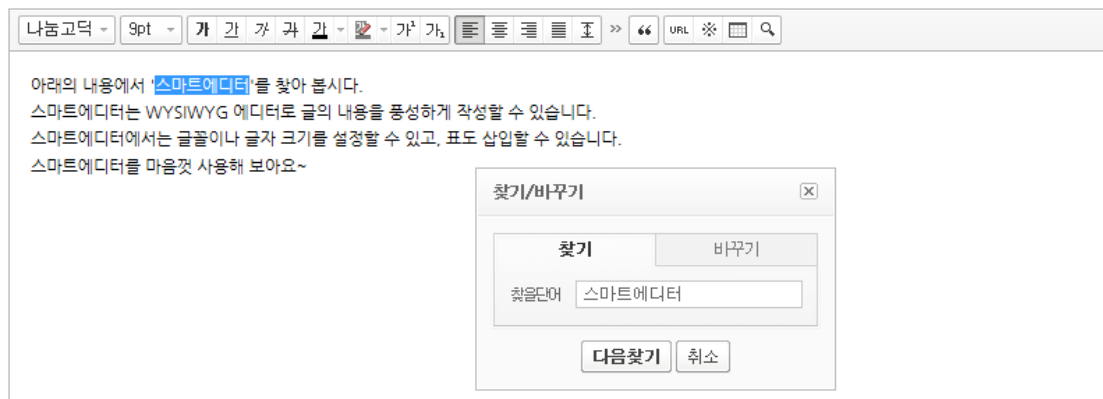


그림 28 단어 찾기 예

다음 그림은 **바꾸기** 탭의 **모두바꾸기**를 클릭하여 '스마트에디터'를 모두 'SmartEditor'로 바꾼 예이다.

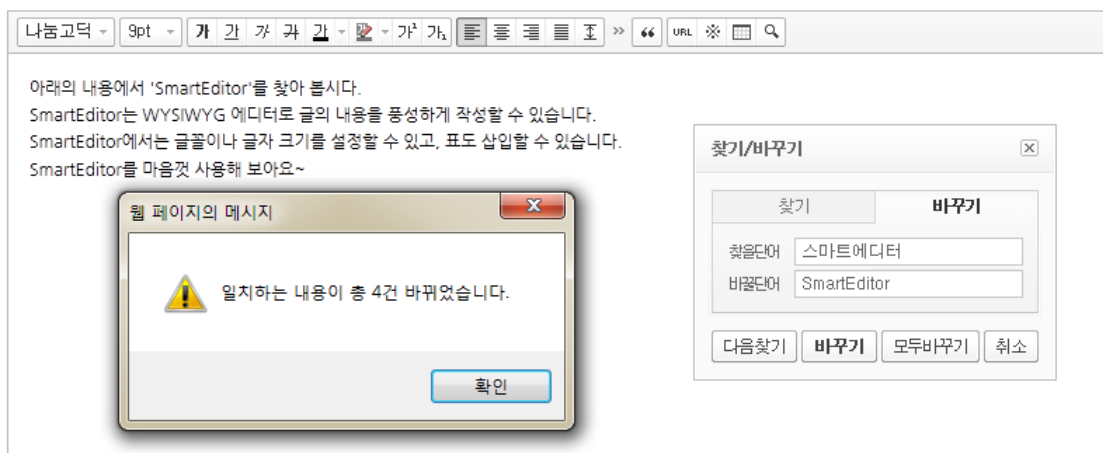


그림 29 단어를 찾아 모두 바꾸기 예

찾기와 바꾸기 기능은 단축키로 이용할 수도 있다. 단축키는 다음과 같다.

표 6 찾기와 바꾸기의 단축키

기능	단축키
찾기	Ctrl+F
바꾸기	Ctrl+H

## 편집 모드

SmartEditor Basic 2.0은 Editor 모드, HTML 모드, TEXT 모드의 세 가지 편집 모드를 제공한다. 각 모드는 편집 화면 오른쪽 아래의 탭으로 선택할 수 있다.

### Editor 모드

툴바의 기능을 이용하여 글을 편집할 수 있는 WISYWIG 편집 모드이다. 다음 그림은 Editor 모드에서 글을 작성하는 예이다.

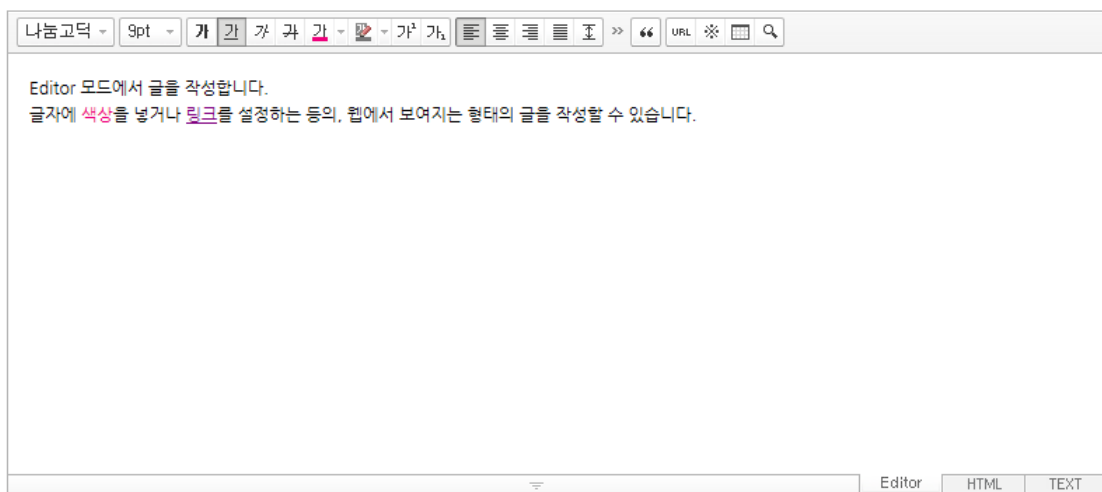


그림 30 Editor 모드에서 글을 작성하는 예

### HTML 모드

HTML 코드를 직접 작성하여 글을 편집할 수 있는 모드이다. 다음 그림은 그림 30에서 작성한 글을 HTML 모드로 변경했을 때의 화면이다. HTML 편집 모드에서는 툴바가 비활성화된다.

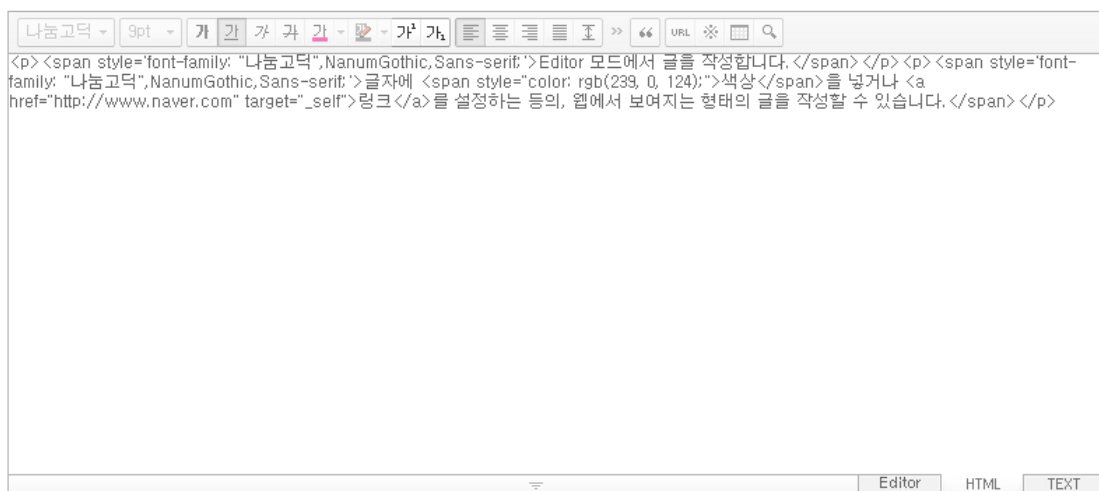


그림 31 HTML 모드에서 글을 작성하는 예

### TEXT 모드

단순 텍스트로만 글을 작성할 수 있다. Editor 모드나 HTML 모드에서 글을 작성하고 TEXT 모드로 변경하면 서식, 이미지 등이 모두 제거되고 텍스트만 남는다. 다음 그림은 그림 30에서 작성한 글을 TEXT 모드로 변경했을 때의 화면이다. TEXT 편집 모드에서는 툴바가 비활성화된다.

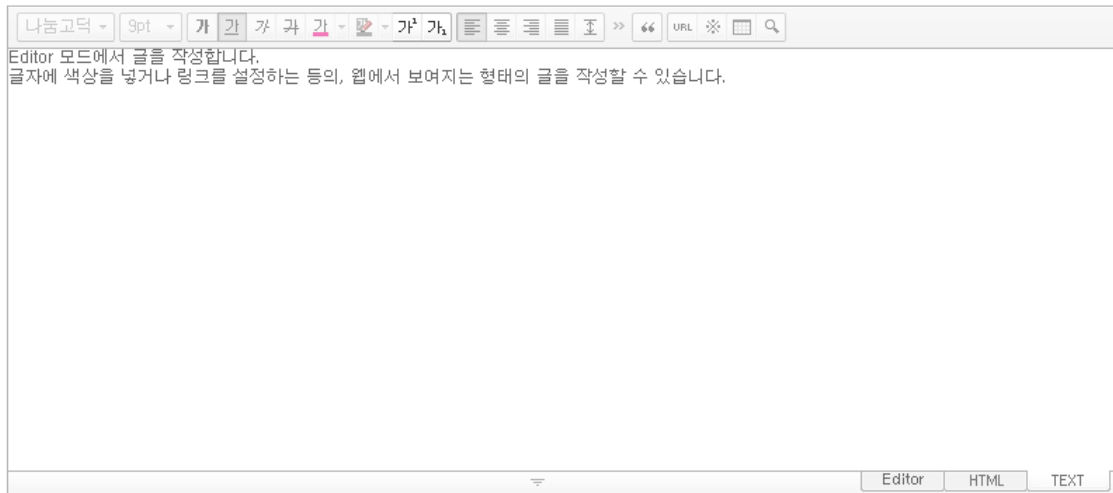


그림 32 TEXT 모드에서 글을 작성하는 예



# 설치하기

이 장에서는 SmartEditor Basic 2.0을 설치하기 전 확인해야 하는 사항을 설명하고, 설치하는 방법과 0.3.x 버전에서 업그레이드하는 방법을 안내한다.

## 설치 전 확인 사항

SmartEditor Basic 2.0을 설치하기 전에 다음과 같은 사항을 확인한다.

### HTML DOCTYPE

SmartEditor Basic 2.0은 HTML Traditional 4.01만 지원하며, Internet Explorer 8.0 이상에서는 Internet Explorer 7.0 호환 모드로 열린다.

### 파일 인코딩

SmartEditor Basic 2.0에서 제공하는 JavaScript, CSS, HTML 코드는 모두 UTF-8 BOM(Byte Order Mark)으로 인코딩되었다.

### 에디터 도메인

에디터를 설치하는 서비스의 도메인과 에디터의 도메인이 일치해야 에디터의 기능이 정상적으로 동작한다. 에디터의 도메인을 설정하는 방법은 "2.0 버전 설치"와 "0.3.x 버전에서 업그레이드"에서 설명한다.

## 다운로드

SmartEditor Basic 2.0을 설치하기 위해 다운로드해야 하는 파일은 다음과 같다.

### 배포 파일 다운로드

<http://dev.naver.com/projects/smarteditor/download>에서 최신 버전을 다운로드한다.

### 배포 파일 구성

SmartEditor Basic 2.0 배포 파일의 기본 구성은 다음과 같다.

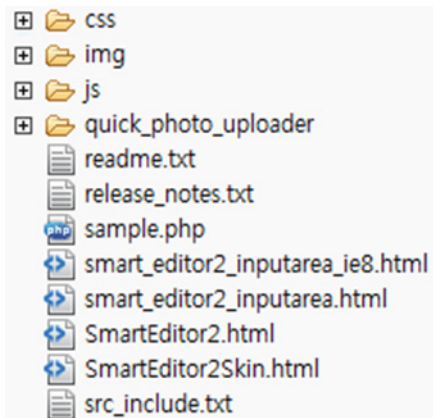


그림 33 배포 파일 구성

각 폴더와 파일의 설명은 다음과 같다.

표 7 배포 파일의 구성과 설명

파일/폴더 이름	설명
/css	에디터에서 사용하는 CSS 파일.
/img	에디터에서 사용하는 이미지 파일.
/js	에디터를 설치할 때 사용하는 JavaScript 파일.
/quick_photo_uploader	에디터의 사진 첨부 기능을 사용하기 위한 팝업 페이지(/popup)와 사진 첨부 기능을 구현한 JavaScript 파일(/plugin).
smart_editor2_inputarea.html	에디터의 편집 영역을 나타내는 HTML 파일. 에디터를 설치할 때 반드시 필요하다.
smart_editor2_inputarea_ie8.html	에디터의 편집 영역을 나타내는 HTML 파일. smart_editor2_inputarea.html 파일과 기능이 동일하며, 브라우저가 Internet Explorer 8.x 이상일 때 사용된다.
SmartEditor2.html	에디터의 데모 페이지. 에디터를 설치할 때 참고할 수 있다.
SmartEditor2Skin.html	에디터를 삽입한 페이지에서 로드하는 에디터의 스킨 HTML 파일. 에디터에서 사용하는 JavaScript 파일과 CSS 파일을 링크하며 에디터의 마크업을 포함한다.
readme.txt	에디터에 대한 간략한 설명이 포함된 문서.
release_notes.txt	릴리즈 시 수정되거나 개선된 사항에 대한 내용이 포함된 문서.
src_include.txt	에디터 원본 소스를 수정하려고 할 때 참고해야 할 가이드 문서.
sample.php	에디터에서 작성한 내용을 view 페이지에서 볼 수 있는 데모.

## 에디터 구조

에디터가 설치되어 있는 SmartEditor2.html 데모 페이지는 다음과 같이 IFRAME으로 에디터 본체를 감싸고 그 안에 IFRAME으로 에디터의 편집 영역을 감싸는 구조이다.

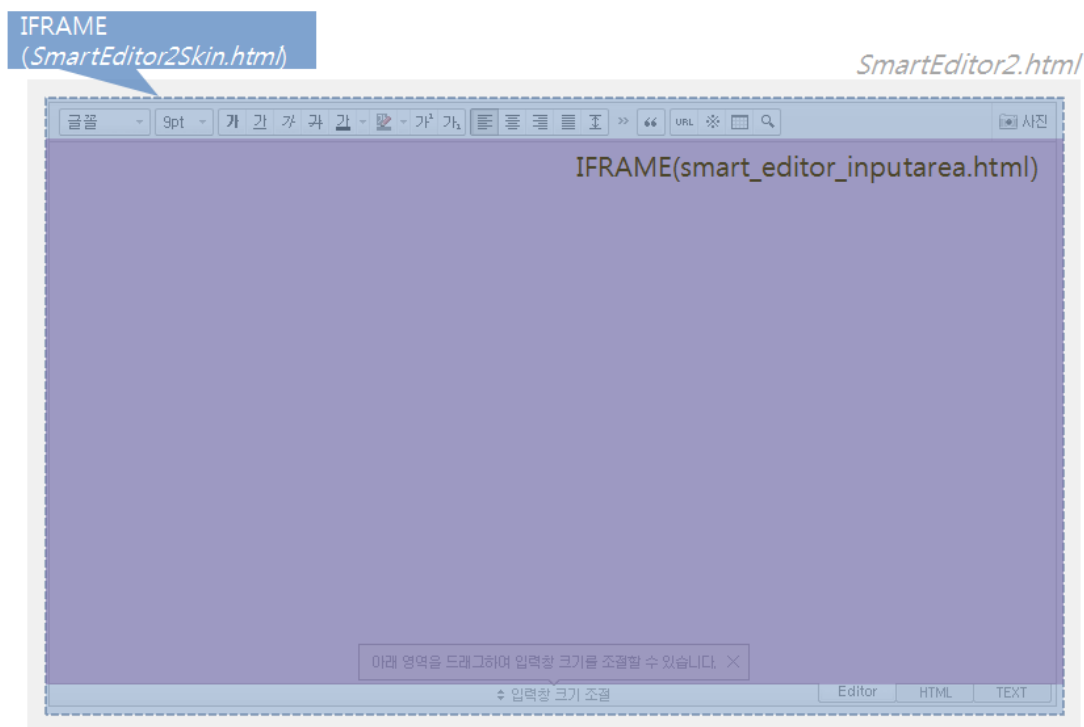


그림 34 SmartEditor2.html의 구조

SmartEditor2.html 파일은 에디터가 삽입된 페이지로, HuskyEZCreator.js 파일의

nhn.husky.EZCreator.createInIFrame() 함수를 호출해 에디터를 생성한다.

nhn.husky.EZCreator.createInIFrame() 함수는 IFRAME을 생성하여 SmartEditor2Skin.html 파일을 로드한다. SmartEditor2Skin.html 파일 로드가 완료되면 에디터를 생성하는 함수가 호출된다.

## 2.0 버전 설치

여기에서는 예제로 pages/write.html 파일에 SmartEditor Basic 2.0을 설치하는 방법을 설명한다("에디터 구조"에서 데모 페이지인 SmartEditor2.html 파일과 같은 역할을 하는 파일이다). SmartEditor Basic 0.3.x 버전이 설치된 상태에서 SmartEditor Basic 2.0 버전으로 업그레이드하려면 "0.3.x 버전에서 업그레이드"를 참고한다.

1. 에디터를 삽입할 페이지와 동일한 도메인에 새 폴더를 생성하고, SmartEditor Basic 2.0 배포 파일에서 다음과 같은 파일과 폴더를 복사해 해당 폴더에 붙여 넣는다. 여기에서 생성할 폴더를 se2라고 하자.

```
/css
/img
/js
smart_editor2_inputarea.html
```

```
smart editor2 inputarea ie8.html
SmartEditor2Skin.html
```

2. write.html 파일에 다음과 같은 내용을 추가한다. 이때 HuskyEZCreator.js 파일의 경로가 맞는지 주의해야 한다.

```
<script type="text/javascript" src="../../se2/js/HuskyEZCreator.js" charset="utf-8"></script>
```

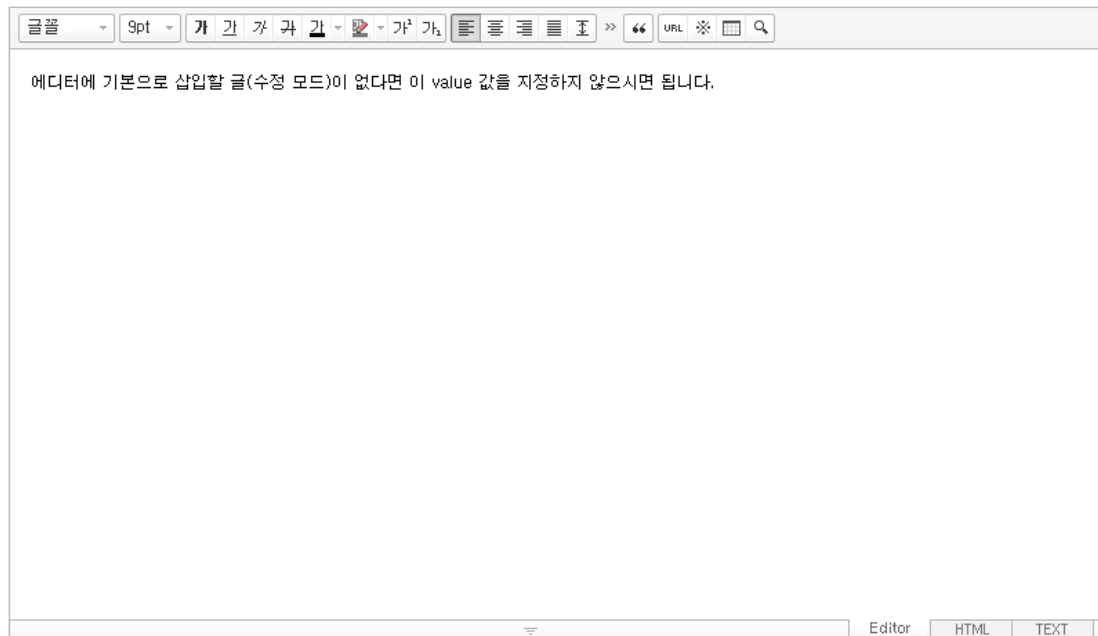
3. write.html 파일에서 에디터를 추가할 위치에 다음과 같이 textarea를 추가한다. 사용자가 에디터에서 작성한 내용은 여기에서 추가한 textarea value를 통해 서버로 전송된다. 기존에 작성하여 저장한 글을 수정하는 경우에는 이 textarea value에 수정할 내용을 지정하여, 에디터가 로드되었을 때 에디터 편집 영역에 기존 글이 표시되도록 한다.

```
<textarea name="irl" id="irl" rows="10" cols="100" style="width:766px; height:412px; display:none;">에디터에 기본으로 삽입할 글(수정 모드)이 없다면 이 value 값을 지정하지 않으시면 됩니다.</textarea>
```

4. write.html 파일에 다음과 같이 에디터를 생성하는 코드를 작성한다. 이때 SmartEditor2Skin.html 파일의 경로가 맞는지 주의해야 한다.

```
<script type="text/javascript">
var oEditors = [];
nhn.husky.EZCreator.createInIFrame({
    oAppRef: oEditors,
    elPlaceHolder: "irl",
    sSkinURI: "../../se2/SmartEditor2Skin.html",
    fCreator: "createSEditor2"
});
</script>
```

5. write.html 파일을 저장하고 에디터가 다음 그림처럼 로드되는지 확인한다.



6. 에디터는 편집 내용을 서버에 전달해 주는 역할만 수행하며, 에디터에서 편집한 내용을 저장하는 작업은 서버 측에서 이루어진다. 에디터는 편집 내용을 전송하기 위해서 먼저 3번 과정에서 추가한 textarea의 value에 편집 내용을 적용한 후에 서버 측 URL에 form을 전송한다.

textarea의 value에 편집 내용을 적용하려면 UPDATE\_CONTENTS\_FIELD 메시지를 호출해야 한다. 이 부분은 다음 write.html 파일의 코드를 참고한다

```
// '저장' 버튼을 누르는 등 저장을 위한 액션을 했을 때 submitContents가 호출된다고 가정한다.
function submitContents(elClickedObj) {
    // 에디터의 내용이 textarea에 적용된다.
    oEditors.getById["ir1"].exec("UPDATE_CONTENTS_FIELD", []);

    // 에디터의 내용에 대한 값 검증은 이곳에서
    // document.getElementById("ir1").value를 이용해서 처리한다.

    try {
        elClickedObj.form.submit();
    } catch(e) {}
}
```

### 0.3.x 버전에서 업그레이드

SmartEditor Basic 0.3.x 버전에서 SmartEditor Basic 2.0으로 업그레이드하는 방법을 설명한다.

1. SmartEditor Basic 0.3.x 버전이 설치된 서비스의 파일 구조가 다음과 같다고 가정한다.

```
/pages
  sample.php -- 에디터의 내용을 저장
  write.html -- 에디터가 설치된 html
/se
  /css          -- 에디터 관련 CSS 파일들
  /img          -- 에디터 관련 이미지 파일들
  /js          -- 에디터 관련 JavaScript 파일들
    Husky.SE_Basic.js
    HuskyEZCreator.js
    jindo.min.js
    SE_CustomPlugins.js
  se_blank.html
  SEditorSkin.html
```

2. 새 폴더를 생성해 SmartEditor Basic 2.0 배포 파일에서 다음과 같은 파일과 폴더를 복사해 해당 폴더에 붙여 넣는다. 여기에서 생성할 폴더를 se2라고 하자.

```
/css
/img
/js
smart_editor2_inputarea.html
smart_editor2_inputarea_ie8.html
SmartEditor2Skin.html
```

3. 기존 write.html 파일을 복사하여 write\_se2.html 파일을 생성한다. 그러면 폴더/파일 구조는 다음과 같다.

```
/pages
  sample.php
  write.html
  write_se2.html      -- write.html 을 복사한 파일
/se
  /css
  /img
  /js
    Husky.SE_Basic.js
    HuskyEZCreator.js
    jindo.min.js
    SE_CustomPlugins.js
  se_blank.html
  SEditorSkin.html
/se2                -- 새로 추가된 폴더
  /css              -- 에디터 관련 CSS들
  /img              -- 에디터 관련 이미지들
  /js               -- 에디터 관련 JavaScript 파일들
  smart_editor2_inputarea.html
  smart_editor2_inputarea_ie8.html
  SmartEditor2Skin.html
```

4. write\_se2.html 파일에서 다음과 같은 CSS 링크를 제거한다.

```
<link href="css/default.css" rel="stylesheet" type="text/css" />
```

5. write\_se2.html 파일에서 HuskyEZCreator.js 파일의 경로를 새 경로로 변경한다.

```
<script type="text/javascript" src="../../se2/js/HuskyEZCreator.js" charset="utf-8"></script>
```

6. write\_se2.html 파일에서 에디터를 추가할 위치에 다음과 같이 textarea가 있는지 확인한다.

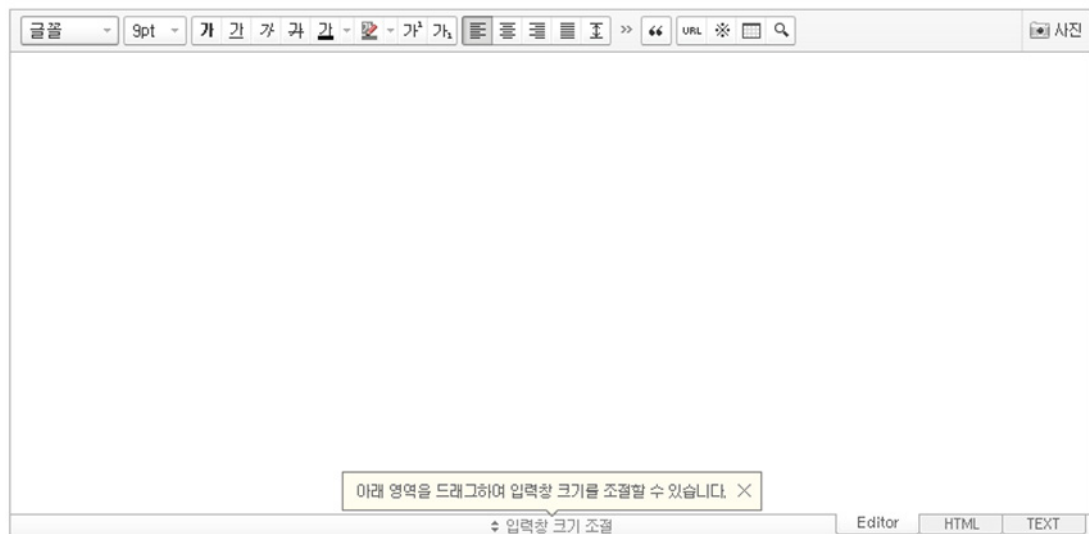
```
<textarea name="irl1" id="irl1" rows="10" cols="100" style="width:766px; height:412px; display:none;"></textarea>
```

7. write\_se2.html 파일에서 에디터를 생성하는 코드의 아래 부분을 다음과 같이 변경한다.

- sSkinURI의 경로를 SEditorSkin.html 파일 경로에서 SmartEditor2Skin.html 파일 경로로 변경한다.
- fCreator의 함수명을 createSEditorInIFrame에서 createSEditor2로 변경한다.

```
<script type="text/javascript">
var oEditors = [];
nhn.husky.EZCreator.createInIFrame({
    oAppRef: oEditors,
    elPlaceholder: "irl1",
    sSkinURI: "../../se2/SmartEditor2Skin.html",
    fCreator: "createSEditor2"
});
</script>
```

8. write.html 파일을 저장하고 에디터가 다음 그림처럼 정상적으로 로드되는지 확인한다.



9. 에디터에서 편집한 내용을 저장하기 위해 textarea에 에디터 편집 영역의 내용을 넣는 코드를 찾아 매개변수를 UPDATE\_IR\_FIELD에서 UPDATE\_CONTENTS\_FIELD로 변경한다.

```
oEditors[0].exec("UPDATE_CONTENTS_FIELD", []); // 에디터의 내용이 textarea에 적용된다.
// document.getElementById("irl1").value 값을 이용해서 에디터에 입력된 내용을 검증한다.
try{
    // 이 라인은 현재 사용 중인 폼에 따라 달라질 수 있다.
    elClicked.form.submit();
}catch(e){}
```

# 기능 추가 및 변경하기

SmartEditor Basic 2.0의 각 기능은 Husky 플러그인을 Husky Core에 등록하여 사용하도록 개발되었다. 따라서 SmartEditor Basic 2.0의 기능을 수정하거나 추가, 삭제하려면 Husky 프레임워크에서 사용하는 용어를 알고 구현 방식을 이해해야 한다.

Husky 프레임워크는 메시지 패싱 방식의 애플리케이션 작성 환경을 제공하는 JavaScript 프레임워크이며, Husky 엔진과 플러그인, 스킨으로 구성되어 있다. 자세한 내용은 "Husky 프레임워크 프로그래밍 가이드"를 참고한다. 이 문서는

<http://dev.naver.com/projects/smarteditor/wiki/HuskyFramework>에서 다운로드할 수 있다.

## 파일과 클래스 명명 규칙

다음은 SmartEditor Basic 2.0에 기능을 추가하거나 기능을 변경하기 위해 Husky 플러그인을 개발할 때 반드시 지켜야 하는 명명 규칙이다.

- Husky 플러그인의 파일 이름은 "hp\_[Husky플러그인이름].js" 형태여야 한다. 예를 들어 기본 에디터 명령어 처리 플러그인인 SE2\_TimeStamper의 파일 이름은 hp\_SE2\_TimeStamper.js이다.
- Husky 플러그인이 아닌 jindo 클래스 또는 함수로 구성된 파일의 이름은 접두사인 "hp\_"를 붙이지 않는다.
- Husky 플러그인 클래스를 생성할 때에는 이름 앞에 nhn.husky를 붙인다.

```
nhn.husky.SE2_TimeStamper = jindo.$Class({
  ...
});
```

- Husky 플러그인의 이름(name)은 중복되지 않는 유일한 값이어야 한다.

```
nhn.husky.SE2_TimeStamper = jindo.$Class({
  name : 'SE2_TimeStamper',
  ...
});
```

- Husky 플러그인 내에서 사용하는 메시지와 메시지 호출로 실행되는 함수인 메시지 핸들러 명명 규칙은 "Husky 프레임워크 프로그래밍 가이드"를 참고한다.

## 기본 클래스 함수 추가 및 변경

기존 클래스에 prototype을 이용하여 함수를 추가하거나 변경하는 방법은 다음과 같다.

### 마크업 변경

마크업이 변경되어서 HTML 요소들을 선언하고 초기화하는 부분을 수정해야 하는 경우에는 다음과 같이 코드를 작성한다.

```
nhn.husky.SE_TimeStamper.prototype._assignHTMLElements = function () {
  this.elInputButton = ...
  this.elInputButton2 = ...
};
```

## 핸들러 추가/변경

특정 클래스에서 동작하는 핸들러를 추가하거나 변경하려면 다음과 같이 코드를 작성한다.

```
nhn.husky.SE2M_AttachFile.prototype.$ON_ATTACHFILE_OPEN_WINDOW = function () {
    var url = ...;
    ...
    window.open(url, ...);
};
```

## 핸들러 제거

특정 클래스의 핸들러가 동작하지 않게 하려면 다음과 같이 코드를 작성한다.

```
nhn.husky.SE2M_AttachFile.prototype.$ON_SET_ATTACH_FILE = function () {};
```

## 기본 기능 삭제

SmartEditor Basic 2.0에서는 사용하지 않는 기능을 삭제하여 코드를 간소화할 수 있다. 여기에서는 툴바에 기본으로 추가되어 있는 인용구 스타일, 하이퍼링크, 특수 기호, 표, 찾기/바꾸기 기능을 삭제하는 예제를 작성한다.

### SE2BasicCreator.js 파일 수정

제거된 기능을 에디터에서 찾지 않도록 삭제할 기능이 등록된 부분을 제거해야 한다.

/js/SE2BasicCreator.js 파일에서 다음과 같이 삭제할 기능이 명시된 줄을 삭제한다.

```
oEditor.registerPlugin(new nhn.husky.SE2M_Quote(elAppContainer)); // 인용구 스타일
oEditor.registerPlugin(new nhn.husky.SE2M_Hyperlink(elAppContainer)); // 하이퍼링크
oEditor.registerPlugin(new nhn.husky.SE2M_SCharacter(elAppContainer)); // 특수 문자
oEditor.registerPlugin(new nhn.husky.SE2M_FindReplacePlugin(elAppContainer)); // 찾기/바꾸기
oEditor.registerPlugin(new nhn.husky.SE2M_TableCreator(elAppContainer)); // 테이블 생성
oEditor.registerPlugin(new nhn.husky.SE2M_TableEditor(elAppContainer)); // 테이블 편집
oEditor.registerPlugin(new nhn.husky.SE2M_TableBlockStyler(elAppContainer)); // 테이블 스타일
```

### SmartEditor2Skin.html 파일 수정

SmartEditor2Skin.html 파일에서 사용하지 않는 버튼에 해당하는 마크업을 지운다. 각각의 기능들은 다음 예처럼 li 태그로 구성되어 있다. li 태그 내부의 태그도 모두 삭제한다.

```
<li class="husky_seditor_ui_quote">
  <button type="button" title="인용구" class="se2_blockquote"><span>인용구</span></button>
  <!-- 인용구 -->
  <div class="se2_layer husky_seditor_blockquote_layer" ...>
    ...
  </div>
  <!-- //인용구 -->
</li>
```

같은 방법으로 하이퍼링크, 특수 기호, 표, 찾기/바꾸기 기능의 마크업을 모두 삭제하면 <ul class="extra">...</ul> 안의 모든 기능이 삭제되므로 ul 태그를 남길 필요가 없다. 따라서 ul 태그까지 삭제한다.



## 툴바 버튼의 위치 변경과 버튼 제거

2.2.1 이상 버전에서는 툴바의 UI를 변경할 수 있다. 2.2.1 미만 버전에서는 툴바에서 버튼의 위치에 따라 버튼이 라운드형이나 사각형으로 표시되어, 버튼의 위치를 변경하거나 버튼을 제거하려면 버튼 이미지를 수정해야 했다. 2.2.1 이상 버전에서는 이런 경우에 이미지를 수정할 필요가 없도록 개선했다.

여기에서는 툴바의 버튼 구조와 툴바 버튼 라운딩 처리 규칙을 설명하고, 툴바 버튼의 위치를 변경하거나 버튼을 제거하는 방법을 설명한다. 툴바에 새로운 UI를 추가하려면 "새로운 기능 추가"를 참고한다.

### 툴바 버튼 구조

툴바 버튼의 기본 HTML 구조는 다음과 같다.

```
<ul>
  <li><button type="button"><span class="_buttonRound">버튼이름</span></button></li>
</ul>
```

툴바는 하나 이상의 버튼이 포함된 버튼 그룹으로 구성된다. HTML 코드에서 버튼 그룹은 ul 요소로 구현하고 그룹 내의 각 버튼은 ul 요소 내의 li 요소로 구현한다. 다음 그림에서 각각의 빨간색 사각형이 버튼 그룹을 나타낸다.



그림 35 툴바의 버튼 그룹

### 툴바 버튼 라운딩 처리 규칙

툴바의 버튼을 표시할 때에는 에디터 내에 존재하는 라운딩 처리용 이미지를 배경으로 사각형의 버튼 이미지를 올린다. 툴바에서 버튼을 라운딩 처리하려면 버튼에 클래스가 "\_buttonRound"인 span 태그를 적용해야 한다. 툴바에 기본적으로 포함된 모든 버튼에는 이 규칙이 적용되어 있다. 새로운 버튼을 추가하는 경우에도 이 규칙을 지켜야 한다.

툴바 그룹 구성에 따라 툴바 버튼은 기본형, 단독형, 더보기로 구성된다. 각 버튼에 대한 설명은 다음과 같다.

- 기본형: 그림 35에서 2, 3, 6번과 같이 버튼 그룹에 여러 개의 버튼이 있는 경우에 해당한다. 버튼 그룹에서 첫 번째 버튼과 마지막 버튼에는 반드시 클래스가 "\_buttonRound"인 span 태그를 적용해야 한다. 예를 들어 그림 35에서 6번 그룹의 가장 마지막 버튼인 찾기/바꾸기 버튼을 삽입하는 코드는 다음과 같다.

```
<li class="husky seditor ui findAndReplace">
  <button type="button" title="찾기/바꾸기" class="se2_find">
    <span class="_buttonRound">찾기/바꾸기</span>
  </button>
  <!-- 찾기/바꾸기 -->
  ... 찾기/바꾸기 레이어 HTML 코드 ...
  <!-- //찾기/바꾸기 -->
</li>
```

```
</ul>
```

- 단독형: 그림 35에서 5번과 같이 버튼 그룹에 버튼이 하나만 있는 경우에 해당한다. 이 경우에도 반드시 클래스가 "\_buttonRound"인 span 태그를 적용해야 한다.

```
<ul>
<li class="husky_seditor_ui_quote">
<button type="button" title="인용구" class="se2_blockquote">
<span class="_buttonRound">인용구</span>
</button>
<!-- 인용구 -->
... 인용구 레이어 HTML 코드 ...
<!-- //인용구 -->
</li>
</ul>
```

- 더보기: 그림 35에서 4번과 같은 경우에 해당한다. 더보기 버튼은 라운딩 처리할 필요가 없으나, 다음과 같이 더보기 버튼을 클릭했을 때 노출되는 레이어 내부의 버튼 그룹은 기본형과 동일하게 라운딩 처리해야 한다.



#### 참고

그림 35에서 1번(글꼴, 글자 크기)과 7번(사진)은 라운딩 처리 대상이 아니다.

## 툴바 버튼 위치 변경하기

앞의 "툴바 버튼 구조"와 "툴바 버튼 라운딩 처리 규칙"의 내용을 참고하여 툴바의 마크업을 변경하면 툴바 버튼의 위치를 변경할 수 있다.

예를 들어 위 첨자, 아래 첨자 버튼을 왼쪽 정렬 버튼 앞으로 이동해 보자. 변경 전 코드는 다음과 같다.

```
<!-- 배경 색 버튼 -->
<li class="se2_pair husky_seditor_ui_BGColor"><span class="selected_color
husky se2m BGColor lastUsed" style="background-color:#4477f9"></span><span
class="husky_seditor_ui_BGColorA"><button type="button" title="배경색"
class="se2_bgcolor"><span>배경색</span></button></span><span
class="husky_seditor_ui_BGColorB"><button type="button" title="더보기"
class="se2_bgcolor_more"><span class="_buttonRound">더보기</span></button></span>
<!-- 배경 색 -->
... 배경 색 레이어 HTML 코드 ...
<!-- //배경 색 -->
</li>
<!-- 위 첨자 버튼 -->
<li class="husky_seditor_ui_superscript"><button type="button" title="윗첨자"
class="se2_sup"><span class="_buttonRound">윗첨자</span></button></li>
<!-- 아래 첨자 버튼 -->
<li class="husky_seditor_ui_subscript"><button type="button" title="아래첨자"
class="se2_sub"><span class="_buttonRound">아래첨자</span></button></li>
</ul>
<ul>
<!-- 왼쪽 정렬 버튼 -->
<li class="husky_seditor_ui_justifyleft"><button type="button" title="왼쪽정렬"
class="se2_left"><span class="_buttonRound">왼쪽정렬</span></button></li>
```



```

        </div>
    </div>
</div>
<!-- //링크 -->
</li>
<!-- 특수 기호 버튼 -->
<li class="husky_seditor_ui_sCharacter">
    <button type="button" title="특수기호" class="se2_character">
        <span class="buttonRound">특수기호</span></button>
    <!-- 특수 기호 -->

```

위 코드에서 링크 버튼에 해당하는 li 요소를 삭제한 코드는 다음과 같다.

```
<ul>
  <!-- 특수 기호 버튼 -->
  <li class="husky_seditor_ui_sCharacter">
    <button type="button" title="특수기호" class="se2_character">
      <span class="_buttonRound">특수기호</span></button>
    <!-- 특수 기호 -->
  </li>
</ul>
```

링크 버튼이 버튼 그룹에서 첫 번째 버튼이었으므로, 링크 버튼이 제거되면 특수 기호 버튼이 버튼 그룹의 첫 번째 버튼이 된다. 따라서 특수 기호 버튼의 라운딩 처리가 누락되지 않았는지 확인해야 한다.

위와 같이 코드를 변경하고 저장하면 다음 그림과 같이 툴바가 변경된다.



### 그림 37 툴바 버튼의 제거

## 참고

그림 35에서 1번(글꼴, 글자 크기) 버튼은 제거하면 안 된다.

## 새로운 기능 추가

여기에서는 SmartEditor Basic 2.0에 없는 기능을 수행하는 플러그인 생성 방법을 설명한다. 예제로 에디터 위쪽의 툴바에 버튼을 추가하고 SE2\_TimeStamper라는 플러그인을 생성하여, 버튼을 클릭하면 에디터에 액티브 레이어가 생성되도록 구현할 것이다.

여러 개의 파일을 직접 링크하면 성능이 저하될 수 있으므로, 여러 개의 JavaScript 파일을 하나의 JavaScript 파일로 묶는 유틸리티를 이용하여 SE2\_Plus.js와 같은 새로운 에디터 파일을 만들어 사용하는 것이 좋다.

## UI 추가

새로운 기능에 필요한 UI를 추가하는 방법은 다음과 같다.

## 이미지 추가

툴바에 삽입할 버튼으로 사용할 이미지 파일(21 x 21 px의 PNG 파일)은 /img 폴더에 저장한다. 해당 버튼 이미지를 사용하도록 다음과 같은 내용을 /css/smart\_editor2.css 파일에 추가한다.

```
#smart_editor2 .se2_text_tool .se2_clock { background:url("../img/clock.png") no-repeat };
```

툴바 버튼을 삽입하는 코드를 작성한다. 다른 SmartEditor Basic 2.0 플러그인으로 사용하려면 다음과 같은 마크업 규칙을 준수해야 한다.

- 버튼은 li 요소 안에 있어야 하며, li > button > span의 구조를 이루어야 한다.
- 버튼을 추가하는 li 태그의 클래스 이름에는 husky\_seditor\_ui 가 앞에 붙어야 한다.

다음은 위 규칙에 따라 새로운 버튼의 HTML 코드를 작성한 예이다.

```
<li class="husky_seditor_ui_TimeStamper">
  <!-- 버튼 -->
  <button type="button" title="현재 시간 삽입" class="se2_clock"><span>시간</span></button>
</li>
```

2.2.1 이상 버전을 사용하는 경우 다음 HTML 코드와 같이 span 태그의 클래스를 "\_buttonRound"로 지정해야 한다.

```
<li class="husky_seditor_ui_TimeStamper">
  <!-- 버튼 -->
  <button type="button" title="현재 시간 삽입" class="se2_clock"><span
class="_buttonRound">시간</span></button>
</li>
```

이 클래스는 라운딩 처리될 버튼을 표시한다. 이에 대한 자세한 내용은 "툴바 버튼 라운딩 처리 규칙"을 참고한다.

## 레이어 추가

버튼을 추가한 위 HTML 코드에 button 태그 다음에 버튼을 클릭했을 때 화면에 표시할 레이어를 추가한다(버튼을 클릭했을 때 레이어가 나타나지 않는 경우에는 이 부분을 생략한다). 이때 div 태그의 클래스에는 se2\_layer라는 구분자와 해당 레이어의 구분자(husky\_seditor\_TimeStamper\_layer)를 할당한다. display 속성의 값은 none으로 설정한다.

```
<li class="husky_seditor_ui_TimeStamper">
  <!-- 버튼 -->
  <button type="button" title="현재 시간 삽입" class="se2_clock"><span
class="_buttonRound">시간</span></button>
  <!-- 레이어 -->
  <div class="se2_layer husky_seditor_TimeStamper_layer"
    style="display:none;margin-left: -198px;">
    <div style="margin: 10px 10px 10px 10px">
      <input type="button" value="Insert Date" class="se_button_time">
    </div>
  </div>
</li>
```

위에서 작성한 코드를 SmartEditor2Skin.html 파일에 삽입한다. 툴바를 구성하는 div의 클래스는 "husky\_seditor\_text\_tool"이다. div 요소 안에서 버튼을 추가하고 싶은 위치의 ul 요소 안에 위 코드를

삽입하거나 새로운 ul 태그를 추가하고 그 안에 삽입한다. 단, "se2\_font\_type" 클래스를 갖는 ul 태그 안에는 추가하면 안 된다. 여기에서는 찾기/바꾸기 버튼 뒤에 삽입한다.

```
<div class="se2 text tool husky seditor text tool">
  <ul class="se2_font_type">
    ... 이 부분에는 버튼을 추가하지 않는다 ...
  </ul>
  <ul>
    ... 생략 ...
    <li class="husky_seditor_ui_findAndReplace">
      ... 생략 ...
    </li>
  </ul>
  <!-- 새로운 버튼 추가 -->
  <ul>
    <li class="husky_seditor_ui_TimeStamper">
      <!-- 버튼 -->
      <button type="button" title="현재 시간 삽입" class="se2_clock"><span
class="_buttonRound">시간</span></button>
      <!-- 레이어 -->
      <div class="se2_layer husky_seditor_TimeStamper_layer"
style="display:none;margin-left: -198px;">
        <div style="margin: 10px 10px 10px 10px">
          <input type="button" value="Insert Date" class="se_button_time">
        </div>
      </div>
    </li>
  </ul>
  <!-- //새로운 버튼 추가 -->
  ... 생략 ...
</div>
```

위와 같이 코드를 작성했을 때 에디터 툴바의 모습은 다음과 같다.



그림 38 버튼을 추가한 에디터 툴바 화면

## 플러그인 등록

플러그인을 제작하기 전에 다음과 같이 SE\_TimeStamper 플러그인을 명시해야 한다.

### SmartEditor2Skin.htm 파일에 JavaScript 파일 경로 추가

SmartEditor2Skin.html 파일을 열어서 플러그인 파일 경로를 추가한다. 파일을 추가할 위치를 결정하기 어렵다면, smarteditor2.min.js를 선언하는 줄 아래에 추가하는 것을 권장한다.

```
<script type="text/javascript" src="js/hp_SE_TimeStamper.js" charset="utf-8"> </script>
```

### SE2BasicCreator.js 파일에 플러그인 등록

/js/SE2BasicCreator.js 파일을 열어서 플러그인을 등록한다. 제일 아래에 추가하는 것을 권장한다.

```
oEditor.registerPlugin(new nhn.husky.SE_TimeStamper(elAppContainer));
```

## 플러그인 제작

플러그인을 제작하려면 핸들러를 추가해야 한다. 여기에서는 Husky 프레임워크 기본 핸들러와 사용자 핸들러를 추가하는 방법을 설명한다.

### Husky 프레임워크 기본 핸들러 추가

"파일과 클래스 명명 규칙"에서 설명한 규칙에 따라 플러그인 파일을 생성하고 다음과 같이 기본 핸들러를 추가한다.

- \$init

플러그인을 생성하면서 동시에 초기화를 수행하려면 \$init 함수에서 초기화를 수행한다.

```
$init : function(elAppContainer){
    this._assignHTMLObjects(elAppContainer);
},
```

- \_assignHTMLElements

지속적으로 사용할 HTML 요소는 \_assignHTMLElements 함수에서 초기화한다. 단, 성능에 많은 영향을 준다면 필요한 시점에 HTML 요소를 초기화하여 사용하는 것이 좋다.

다음은 툴바에 추가한 버튼을 클릭하면 보여줄 div 레이어를 초기화하는 코드이다.

```
_assignHTMLObjects : function(elAppContainer){
    this.oDropdownLayer =
        cssquery.getSingle("DIV.husky_seditor_TimeStamper_layer", elAppContainer);
},
```

- \$ON\_MSG\_APP\_READY

모든 플러그인이 등록된 이후에 초기화를 수행해야 하는 경우에는 \$ON\_MSG\_APP\_READY에서 처리한다. 이 메시지는 에디터 초기화에 영향을 주므로 HTML 요소를 초기화하는 작업과 HTML 요소에 이벤트를 연결하는 작업만 수행한다.

다음은 버튼에 이벤트를 연결하는 코드이다. 이때 REGISTER\_UI\_EVENT에 전달되는 3개의 매개변수는 각각 '추가한 버튼의 클래스 이름에서 husky\_seditor\_ui를 제외한 부분', '발생할 이벤트', '이벤트가 발생하면 호출될 핸들러'를 의미한다.

```
$ON_MSG_APP_READY : function() {
    this.oApp.exec('REGISTER_UI_EVENT', ['TimeStamper', 'click',
        'SE TOGGLE TIMESTAMPER LAYER']);
},
```

### 사용자 핸들러 추가

Husky 플러그인에서 제공하는 핸들러 외에 사용자 핸들러도 생성해야 한다. 핸들러 이름 앞에는 \$ON\_을 붙여야 하며, 핸들러 이름 작성 시 권장 사항은 다음과 같다.

- 대문자로 작성한다.
- 그대로 전달되는 브라우저 이벤트에 대한 핸들러의 이름은 다음과 같이 EVENT\_로 시작한다.

```
$ON_EVENT_EDITING_AREA_KEYUP
```

- 반드시 특정 작업을 수행해야 하는 것이 아니라 특정 이벤트가 발생했음을 통지받는 핸들러의 이름은 다음과 같이 MSG\_로 시작한다.

```
$ON_MSG_EDITOR_READY
```

앞에서 TimeStamper 플러그인 예제에 툴바 버튼에 클릭 이벤트를 추가하고, 버튼을 클릭하면 호출될 핸들러를 선언했다. 여기에서는 호출될 핸들러인 SE\_TOGGLE\_TIMESTAMPER\_LAYER를 작성한다. 함수 안에는 버튼을 클릭하면 수행할 동작을 작성한다.

```
$ON_SE_TOGGLE_TIMESTAMPER_LAYER : function(){
    this.oApp.exec("TOGGLE_TOOLBAR_ACTIVE_LAYER", [this.oDropdownLayer]);
}
```

SE\_TOGGLE\_TIMESTAMPER\_LAYER 핸들러 함수 안에서 호출하는 TOGGLE\_TOOLBAR\_ACTIVE\_LAYER 핸들러는 에디터에서 제공하는 핸들러로, 두 번째 매개변수인 this.oDropdownLayer를 액티브 레이어 그룹에 추가하고 툴바 아래에 다음과 같은 레이어를 생성한다.



그림 39 액티브 레이어를 추가한 에디터 화면

이제 Insert Date 버튼을 누르면 본문에 시간이 삽입되는 함수를 작성한다. 함수의 이름은 \$ON\_PASTE\_NOW\_DATE라고 하자. 이 함수를 작성하는 방법은 다음과 같다.

1. 레이어 안의 버튼을 \_assignHTMLObjects에서 초기화한다.
2. \$ON\_MSG\_APP\_READY에서 초기화한 버튼에 이벤트를 할당한다.
3. PASTE\_NOW\_DATE 핸들러를 작성한다. 이 핸들러 안에서는 에디터에서 제공하는 PASTE\_HTML이라는 핸들러를 호출한다. PASTE\_HTML은 본문에 콘텐츠를 넣을 때 사용되는 핸들러이다. 두 번째 매개변수로 본문에 삽입할 내용을 입력한다.

```
$ON PASTE NOW DATE : function(){
    this.oApp.exec("PASTE HTML", [new Date()]);
}
```

다음은 플러그인을 생성을 완료하고, 버튼을 클릭하여 본문에 현재 시각을 넣은 화면이다.



그림 40 TimeStamper 실행한 에디터 화면

## SE\_TimeStamper 플러그인 전체 코드

지금까지 툴바에 새로운 기능을 추가하는 방법을 간단하게 알아보았다. hp\_SE\_TimeStamper.js의 전체 코드는 다음과 같다.

```
nhn.husky.SE_TimeStamper = jindo.$Class({
    name : "SE_TimeStamper",
    $init : function(elAppContainer){
        this._assignHTMLObjects(elAppContainer);
    }
});
```



```
},  
  
assignHTMLObjects : function(elAppContainer){  
    this.oDropdownLayer =  
        cssquery.getSingle("DIV.husky_seditor_TimeStamper_layer", elAppContainer);  
    //div 레이어안에 있는 input button을 cssquery로 찾는 부분.  
    this.oInputButton = cssquery.getSingle(".se_button_time", elAppContainer);  
},  
  
$ON MSG APP READY : function(){  
    this.oApp.exec("REGISTER_UI_EVENT",  
        ["TimeStamper", "click", "SE_TOGGLE_TIMESTAMP_LAYER"]);  
    //input button에 click 이벤트를 할당.  
    this.oApp.registerBrowserEvent(this.oInputButton, 'click', 'PASTE_NOW_DATE');  
},  
  
$ON SE TOGGLE TIMESTAMPER LAYER : function(){  
    this.oApp.exec("TOGGLE TOOLBAR ACTIVE LAYER", [this.oDropdownLayer]);  
},  
  
$ON_PASTE_NOW_DATE : function(){  
    this.oApp.exec("PASTE_HTML", [new Date()]);  
}  
});
```

# 사진 퀵 업로더

이 장에서는 사진 퀵 업로더를 소개하고, SmartEditor Basic 2.0에 사진 퀵 업로더 기능을 설치하고 제대로 설치되었는지 확인하는 방법을 설명한다.

## 소개

사진 퀵 업로더는 SmartEditor Basic 2.0을 이용하여 글을 작성할 때 사진을 쉽고 빠르게 업로드할 수 있는 팝업 UI와 에디터 플러그인을 제공한다. 에디터의 팝업 관련 코드뿐만 아니라 파일을 업로드하는 서버 작업도 구현해야 한다. 팝업 UI에서 서비스와 연결되는 방법은 이 문서에서 다루지 않는다.

사용자 브라우저의 HTML5 지원 여부에 따라 다음 두 가지의 UI가 사용된다.

### HTML5 지원 브라우저

HTML5를 지원하는 브라우저에서는 한 장 이상의 사진을 끌어다 지정된 영역에 놓으면 사진이 첨부된다.



### 그림 41 HTML5 지원 브라우저의 사진 퀵 업로더

HTML5 지원 브라우저의 사진 퀵 업로더는 HTML5의 드래그 앤드 드롭과 File API를 사용한다. 2012년 12월을 기준으로 Internet Explorer 10 이상 버전, Firefox 15 이상 버전, Chrome 22 이상 버전, Safari 5.1 이상 버전이 HTML5의 드래그 앤드 드롭과 File API를 지원한다. 드래그 앤드 드롭과 File API를 지원하는 브라우저에 대한 자세한 정보는 다음 주소를 참고한다.

- 드래그 앤드 드롭: [http://caniuse.com/#search=file api](http://caniuse.com/#search=file%20api)
- File API: [http://caniuse.com/#search=drag and drop](http://caniuse.com/#search=drag%20and%20drop)

### HTML5 미지원 브라우저

HTML5를 지원하지 않는 브라우저에서는 한 번에 한 장의 사진만 업로드할 수 있다. 페이지를 전환하지 않고 사진을 업로드할 수 있으며, 바로 성공 여부를 알려준다.



### 그림 42 HTML5 미지원 브라우저의 사진 퀵 업로더

HTML5 미지원 브라우저의 사진 퀵 업로더는 NHN에서 제작된 JavaScript 라이브러리인 Jindo의 FileUploader를 이용하여 제작한 컴포넌트이다. Jindo FileUploader에 대한 자세한 내용은 "부록. jindo.FileUploader"을 참고한다. 그 외에 Jindo에 대한 자세한 내용은 <http://dev.naver.com/projects/jindo>에서 확인할 수 있다.

## 파일 구성

사진 퀵 업로더를 설치할 때 필요한 파일은 다음과 같다.

### 팝업 HTML 및 JavaScript

SmartEditor Basic 2.0 배포 파일의 /quick\_photo\_uploader/popup 디렉터리에는 다음과 같이 사진 퀵 업로더의 팝업 페이지를 구성하는 HTML 파일과 JavaScript 파일이 포함되어 있다.

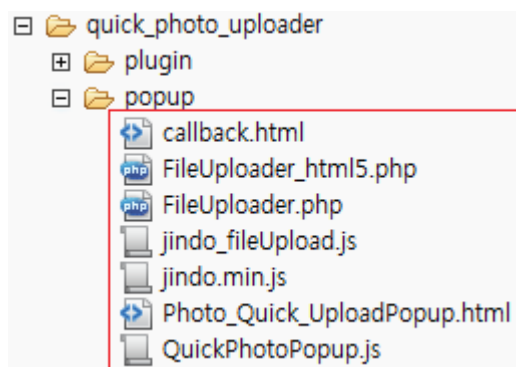


그림 43 사진 퀵 업로더 팝업 HTML 및 JavaScript 파일 구성

각 파일의 설명은 다음과 같다.

표 8 제공되는 파일 설명

구분	설명
callback.html	Ajax 통신용 파일. Jindo fileUpload 컴포넌트를 이용하여 파일을 업로드할 때, 페이지 전환 없이 결과를 전달받을 수 있도록 한다.
FileUploader.php	PHP 서버에서 사용자로부터 전송된 파일을 저장하고 파일 정보를 사용자에게 내려주는 코드. HTML5 미지원 브라우저에서 사용한다.
FileUploader_html5.php	PHP 서버에서 사용자로부터 전송된 파일을 저장하고 파일 정보를 사용자에게 내려주는 코드. HTML5 가 지원되는 브라우저에서 사용한다.
Jindo_fileUpload.js	Jindo 컴포넌트 중 fileUpload 에 필요한 코드만 포함하는 파일
jindo.min.js	사진 퀵 업로더의 JavaScript 파일에서 사용하는 JindoJS 파일
Photo_Quick_UploadPopup.html	사진 퀵 업로더에서 사용하는 HTML 팝업 페이지
QuickPhotoPopup.js	Photo_Quick_UploadPopup.html 파일에서 사용하는 JavaScript 파일. 사진 퀵 업로더를 설치할 때 코드를 반드시 수정해야 한다.

## 이미지

SmartEditor Basic 2.0 배포 파일의 \wimg\photoQuickPopup 디렉터리에는 다음과 사진 퀵 업로더 팝업 페이지를 구성하는 이미지 파일이 포함되어 있다.

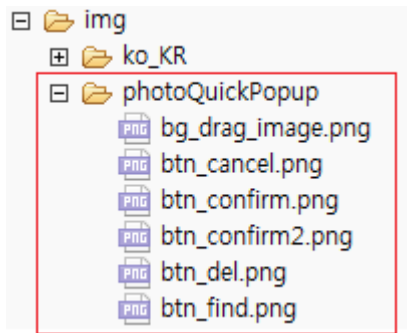


그림 44 사진 퀵 업로더 팝업의 image 구성

사진 퀵 업로더를 설치하려면 위 이미지 파일을 복사하고 JavaScript 파일과 HTML 파일의 이미지 경로를 수정해야 한다.

### 에디터 플러그인

SmartEditor Basic 2.0 배포 파일의 /quick\_photo\_uploader/plugin 디렉터리에는 사진 퀵 업로더 플러그인 파일이 포함되어 있다. 파일 경로 등이 변경되면 hp\_SE2M\_AttachQuickPhoto.js 파일을 수정해야 한다.

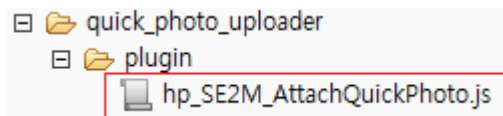


그림 45 사진 퀵 업로더의 plugin 구성

## 설치하기

SmartEditor Basic 2.0에 사진 퀵 업로더를 적용하는 방법을 설명한다. 사진 플러그인을 추가하고 툴바에 사진 버튼을 추가한 후 팝업 JavaScript를 수정한다. 다운로드한 SmartEditor Basic 2.0 툴바에 **사진** 아이콘이 있다면 "PHP 서버 설정"부터 참고한다.

### 사진 플러그인 추가

1. js 폴더의 SE2BasicCreator.js 파일에 다음과 같이 플러그인을 추가한다.

```
oEditor.registerPlugin(new nhn.husky.SE2M_AttachQuickPhoto(elAppContainer)); // 사진
```

2. SmartEditor2Skin.html 파일에 다음과 같이 JavaScript 파일을 선언한다.

```
<script type="text/javascript"
src="./quick photo uploader/plugin/hp SE2M AttachQuickPhoto.js" charset="utf-8">
</script>
```

3. hp\_SE2M\_AttachQuickPhoto.js 파일에서 makePopupURL() 함수를 수정한다. sPopupUrl에는 사진 퀵 업로더 팝업 HTML 파일의 경로를 입력한다.

```
/**
 * 서비스별로 팝업에 parameter를 추가하여 URL을 생성하는 함수
 */
makePopupURL : function() {
```

```
var sPopupUrl = "../quick_photo_uploader/popup /Photo_Quick_UploadPopup.html";
return sPopupUrl;
},
```

## 툴바에 사진 버튼 추가

에디터 마크업이 포함된 HTML 파일에 다음과 같이 마크업을 추가한다. CSS와 이미지는 SmartEditor Basic 2.0.0 이상 버전에서 포함된다.

```
<div class="se2 tool" id="se2 tool">
  <div class="se2_text_tool husky_seditor_text_tool">
    // (생략)
  </div> 태그가 연속으로 2번 나오기 전에 아래 내용을 추가한다.
  <!-- icon toolbar -->
  <ul class="se2_multy">
    <li class="se2_mn husky_seditor_ui_photo_attach">
      <button type="button" class="se2_photo_ico_btn">
        <span class="se2_icon"></span><span class="se2_mntxt">사진<span
class="se2_new"></span></span>
      </button>
    </li>
  </ul>
</div>
</div>
```

## PHP 서버 설정

SmartEditor Basic 2.0에서는 사용자 PC의 사진 퀵 업로드 팝업에서 서버로 파일을 업로드하는 코드까지 제공한다. 이 코드는 PHP로 구현되어 있으며 이 문서에서는 오픈 소스인 Apache 서버와 PHP 서버를 개인 PC에 설치한다고 가정하고 사진 퀵 업로더 설치 방법을 설명한다. Apache 서버와 PHP 서버를 설치하는 방법과 다른 종류의 서버를 사용하는 방법은 이 문서에서 다루지 않는다.

### 참고

초보자에게는 복잡한 설정이 필요 없는 APM(Apache + PHP + MySQL) 설치를 권장한다. 간단하게 APM을 설치하려면 <http://www.apmsetup.com/download.php>를 참고한다.

## 파일을 업로드할 폴더 생성

서버를 설치한 후에는 파일을 업로드할 폴더를 생성한다. 서버의 기본 root를 변경하지 않으면 C:\Program Files\Apache Software Foundation\Apache2.2\htdocs가 서버의 root 폴더로 설정되어 있다. 서버의 기본 root 폴더 안에 php\_uploader 폴더를 생성하고, 생성한 폴더 안에 upload 폴더와 php 폴더를 생성한다. 사진 퀵 업로더에서 첨부한 사진은 upload 폴더에 저장된다.

## PHP 파일 설정

SmartEditor Basic 2.0에서 제공하는 PHP 파일을 사용하여 PHP 서버를 설정한다. 우선 quick\_photo\_uploader/popup/ 폴더의 FileUploader.php 파일과 FileUploader\_html5.php 파일을 앞에서 생성한 php\_uploader/php 폴더로 이동시킨다. 그리고 FileUploader.php 파일과 FileUploader\_html5.php 파일에서 "http://test.naver.com/" 부분을 실제 사용할 서버의 URL로 변경한다.

만약 "http://test.naver.com"을 수정하지 않은 채로 임시로 테스트하려면,

C:\Windows\System32\drivers\etc\hosts 파일의 마지막에 192.162.10.1 test.naver.com을 입력한다.

이는 서버의 IP 주소가 DNS에 등록되지 않았더라도 host명으로 접근할 수 있게 한다.

- FileUploader.php 파일

```
$url .= "&sFileName=".urlencode(urlencode($name));
// $url .= "&size=". $_FILES['Filedata']['size'];
//아래 URL을 변경하시면 됩니다.
$url .= "&sFileURL=http://test.naver.com/popup/upload/".urlencode(urlencode($name));
```

- FileUploader\_html5.php 파일

```
if(file_put_contents($newPath, $file->content)) {
    $sFileInfo .= "&bNewLine=true";
    $sFileInfo .= "&sFileName=".$file->name;
    $sFileInfo .= "&sFileURL=http://test.naver.com/popup/upload/".$file->name;
}
```

PHP 서버 설정을 완료한 후에는 다음과 같이 파일과 폴더가 구성되어 있어야 한다.

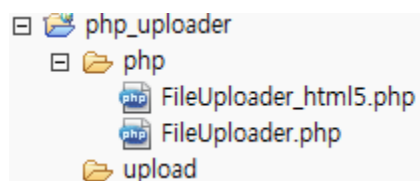


그림 46 사진 퀵 업로더의 PHP 서버 구성

## 팝업 JavaScript 수정

팝업 페이지에 사용되는 JavaScript 코드는 quick\_photo\_uploader/popup/QuickPhotoPopup.js 파일에 포함되어 있다. HTML5를 지원하지 않는 브라우저에서 사용하는 callFileUploader() 함수와 HTML5를 지원하는 브라우저에서 사용하는 html5Upload() 함수, 그리고 공통으로 사용하는 setPhotoToEditor() 함수를 수정해야 한다.

### callFileUploader() 함수

callFileUploader() 함수는 HTML5를 지원하지 않는 브라우저에서 사용되며, Jindo FileUploader 컴포넌트를 초기화한다. 파일 선택이 완료되었을 때, 업로드가 정상적으로 완료되었을 때, 업로드가 실패했을 때 수행할 동작을 이 함수 내에 구현해야 한다. Jindo FileUploader 컴포넌트에 대한 자세한 내용은 "부록. jindo.FileUploader"를 참고한다.

표 9 jindo.FileUploader 구성 요소

구분	설명
sUrl	사진을 업로드할 서버의 경로. "PHP 서버 설정"에서 설정한 경로를 입력한다. http://test.naver.com/php_uploader/php/FileUpload.php
sFiletype	업로드할 수 있는 파일 형식. 각 형식은 세미콜론(;)으로 구분한다. 예: *.jpg;*.png;*.bmp;*.gif

구분	설명
select()	파일 선택이 완료되었을 때 발생하는 함수
success()	업로드가 성공적으로 완료되었을 때 발생하는 함수
error()	업로드가 실패했을 때 발생하는 함수

사진 퀵 업로더를 설치한 후 실제로 사용하려면 반드시 sUrl 속성을 "PHP 서버 설정"에서 설정한 PHP 서버의 PHP 파일 URL로 변경해야 한다.

### html5Upload() 함수

html5Upload() 함수는 HTML5를 지원하는 브라우저에서 사용되며, Jindo \$Ajax로 다음과 같이 구현된다.

```
function html5Upload() {
    var tempFile,
        sUploadURL;

    //업로드할 URL 입력
    sUploadURL= 'http://test.naver.com/php_uploader/php/FileUploader_html5.php';

    //여러 파일이 선택 되었을 때, 파일을 하나씩 보내고 결과를 받음.
    for(var j=0, k=0; j < nImageInfoCnt; j++) {
        tempFile = htImageInfo['img'+j];
        try{
            if(!tempFile){
                //Ajax통신하는 부분. 파일과 업로드 할 url을 전달한다.
                callAjaxForHTML5(tempFile,sUploadURL);
                k += 1;
            }
        }catch(e){}
        tempFile = null;
    }
}

function callAjaxForHTML5 (tempFile, sUploadURL){
    var oAjax = jindo.$Ajax(sUploadURL, {
        type: 'xhr',
        method : "post",
        onload : function(res){ // 요청이 완료되면 실행될 콜백 함수
            if (res.readyState() == 4) {
                //성공 시에 responseText를 가지고 array로 만드는 부분.
                makeArrayFromString(res. response.responseText);
            }
        },
        timeout : 3,
        onerror : jindo.$Fn(onAjaxError, this).bind()
    });
    oAjax.header("contentType","multipart/form-data");
    oAjax.header("file-name",encodeURIComponent(tempFile.name));
    oAjax.header("file-size",tempFile.size);
    oAjax.header("file-Type",tempFile.type);
    oAjax.request(tempFile);
}
```

사진 퀵 업로더를 설치한 후 실제로 사용하려면 반드시 sUrl 속성을 "PHP 서버 설정"에서 설정한 PHP 서버의 PHP 파일 URL로 변경해야 한다.

html5Upload() 함수에서 호출하는 callAjaxForHTML5() 함수는 서버와 통신하는 코드를 분리해 놓은 함수로, 구성 요소는 다음과 같다.



표 10 callAjaxForHTML5() 함수 구성 요소

구분	설명
onload	요청 완료 시 실행되는 콜백 함수. 반드시 지정해야 하며 콜백 함수의 파라미터로 응답 객체인 <a href="#">\$Ajax.Response</a> 객체가 전달된다. 위 예제에서는 onload 에 정의한 함수에서 <code>makeArrayFromString ()</code> 함수를 호출한다.
onerror	오류 발생 시 실행되는 콜백 함수. 생략하면 오류가 발생해도 onload 에 지정한 콜백 함수를 실행한다. 위 예제에서는 <code>onAjaxError</code> 를 호출한다.
htImageInfo	이미지 파일의 정보. <code>drop()</code> 함수에서 데이터를 <code>hashTable</code> 형태로 데이터를 입력하며, 전달되는 <code>dataTransfer.File</code> 은 <code>name</code> , <code>size</code> , <code>type</code> 등의 정보를 제공한다. <code>drop()</code> 함수에서의 이미지 파일 정보 처리 방식을 변경하면 다른 동작에도 영향을 준다. 따라서 서버에서 처리하기 편한 형태로 변경하려면 다음 예와 같이 <code>html5Upload()</code> 함수에서 서버로 데이터를 전송하기 전 단계에서 데이터를 가공하는 것을 권장한다. 예: <code>htImageInfo = {'img1':file1, 'img2':files2, ... };</code>

Jindo \$Ajax는 다양한 통신 방법과 함수를 제공하므로 다음 내용을 참고하여 수정하는 것을 권장한다.

- Jindo \$Ajax: [http://jindo.dev.naver.com/docs/jindo/latest/desktop/ko/classes/\\$Ajax.html](http://jindo.dev.naver.com/docs/jindo/latest/desktop/ko/classes/$Ajax.html)
- HTML5 File API: <http://www.w3.org/TR/FileAPI/>
- Jindo 컴포넌트: <http://dev.naver.com/projects/jindo>

### setPhotoToEditor() 함수

`setPhotoToEditor()` 함수는 브라우저의 HTML5 지원 여부와 상관없이 사용되며, 파일이 성공적으로 업로드되면 서버로부터 받은 데이터를 에디터에 삽입한다. 에디터로 전송되는 기본 데이터 형식은 각 사진의 정보로 구성된 `HashMap`을 포함하는 `Array`이다.

표 11 setPhotoToEditor 함수 구성 요소

구분	설명
sFileName	파일명
sFileURL	도메인과 파일명을 포함한 전체 경로 예: <code>http://test.naver.com/php_uploader/upload/alert.jpg</code>
bNewLine	이미지 사이에 간격( <code>br</code> tag)을 넣을지 여부를 <code>boolean</code> 값으로 지정한다.

`setPhotoToEditor()` 함수를 통해 에디터로 전송되는 데이터의 예는 다음과 같다.

```
var res =[ {sFileName : "코코몽.jpg",
            sFileURL : "http://image_server.domain.net/20120315/코코몽.jpg",
            bNewLine : true },
            {sFileName : "오몽.jpg",
```

```
sFileURL : " http://image_server.domain.net/20120315/오몽.jpg",  
bNewLine : true }, ... ];
```

## 사진 쿼 업로더 사용해 보기

PHP 서버 설치와 팝업 JavaScript 수정을 완료한 후에는 사진 쿼 업로더가 제대로 설치되었는지 다음과 같은 방법으로 확인한다.

1. 사진 쿼 업로더를 설치한 에디터를 열어 메뉴 툴바의 **사진**을 클릭한다.
2. 팝업 페이지가 열리면 원하는 사진 파일을 끌어다 놓거나 **찾아보기**를 클릭한 후 파일을 선택한다.
3. **확인**을 클릭한다.

에디터 내부에 이미지가 정상적으로 삽입되었다면 사진 쿼 업로더가 제대로 설치된 것이다.

## 부록. jindo.FileUploader

이 장에서는 사진 쿼 업로더에서 사용하는 jindo.FileUploader에 대해서 설명한다.

### jindo.FileUploader의 특징

jindo.FileUploader의 특징은 다음과 같다.

- 사용자가 선택할 수 있는 파일 확장자를 지정할 수 있다.
- 사용자가 파일을 선택하는 것과 동시에 파일을 전송한다.
- 파일 전송 성공 여부에 따라 예외 처리가 가능하다.
- 파일은 1개만 선택할 수 있다. 여러 개의 파일을 동시에 선택할 수는 없다.

### 초기화

jindo.FileUploader를 사용하려면 HTML 파일을 다음과 같이 작성한다.

```
<form method="POST" enctype="multipart/form-data">
  <input type="file" id="file_select">
</form>
```

- 전송할 input[type=file]은 반드시 form 요소를 부모 요소로 가져야 한다.
- form의 method 속성 값은 "post", enctype 속성 값은 "multipart/form-data"이어야 한다.

JavaScript 파일은 다음과 같이 작성한다.

```
var oFileUploader = new jindo.FileUploader(jindo.$("file_select"), {
  //업로드할 서버의 URL (Form 전송 대상)
  sUrl : '/samples/response/FileUpload.php',
  //업로드 이후에 IFRAME이 리다이렉트될 콜백 페이지 주소
  sCallback : '/Jindo_Component/FileUploader/callback.html',
  //post할 데이터 셋. 예: { blogId : "testid" }
  htData : {}
});
```

### 요청 수행 과정

jindo.FileUploader가 요청을 수행하는 과정은 다음과 같다.

1. 요청을 받으면 Form 내부에 보이지 않는 IFRAME을 생성한다. 생성된 IFRAME은 지정된 URL로 업로드할 파일을 전송하며, 콜백 페이지 주소(callback)와 콜백 함수의 이름(callback\_func)을 서버에 전달한다.  
예: FileUpload.php?callback=callback.html&callback\_func=tmpFrame\_84101\_func
2. 파일을 전송받은 서버에서는 파일과 콜백 함수의 이름을 콜백 페이지 주소로 리다이렉트한다. 파일 전송에 실패하면 쿼리 스트링으로 errstr=error를 추가한다. 콜백 함수에 추가로 전달할 정보가 있으면 쿼리 스트링으로 추가한다.

3. 콜백 페이지 주소로 리다이렉트된 IFRAME은 부모 요소의 FileUploader 객체에 콜백 함수를 실행시키고 자신을 제거한다.

서버 측 FileUploader.php 파일의 예는 다음과 같다.

```
//기본 리다이렉트
$url = $_REQUEST["callback"] . '?callback_func=' . $_REQUEST["callback_func"];

if (is_uploaded_file($_FILES['Filedata']['tmp_name'])) { //성공 시 파일 사이즈와 URL 전송
    $tmp_name = $_FILES['Filedata']['tmp_name'];
    $name = $_FILES['Filedata']['name'];
    $new_path = "upload/" . urlencode($name);
    @move_uploaded_file($tmp_name, $new_path);
    $url .= "&size=" . $_FILES['Filedata']['size'];
    $url .= "&url=http://test.naver.com/components/upload/" . urlencode(urlencode($name));
} else { //실패시 errstr=error 전송
    $url .= '&errstr=error';
}

header('Location: ' . $url);
```

콜백 페이지 HTML 파일의 내용의 예는 다음과 같다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>FileUploader Callback</title>
</head>
<body>
<script type="text/javascript">
    // document.domain 설정
    document.domain = "nhncorp.com";
    // execute callback script
    var sUrl = document.location.search.substr(1);
    if (sUrl != "blank") {
        var oParameter = {}; // query array
        sUrl.replace(/([^\=]+)=([^\&]*)(&|$)/g, function() {
            oParameter[arguments[1]] = arguments[2];
            return "";
        });
        if ((oParameter.errstr || '').length) { // on error

            (parent.jindo.FileUploader._oCallback[oParameter.callback_func+'_error']) (oParameter);
        } else {

            (parent.jindo.FileUploader.oCallback[oParameter.callback_func+' success']) (oParameter);
        }
    }
</script>
</body>
```

## 커스텀 이벤트

jindo.FileUploader의 커스텀 이벤트는 다음과 같다.

### sUploadURL

파일 선택 완료를 의미한다. sUploadURL 객체의 속성은 다음과 같다.

이름	타입	설명
sType	String	커스텀 이벤트 이름

이름	타입	설명
sValue	String	선택된 file input 의 값
bAllowed	Boolean	선택된 파일의 형식이 허용되는지 여부(값이 false 이면 sMsgNotAllowedExt 에 지정된 메시지를 경고 창에 표시)
sMsgNotAllowedExt	String	선택된 파일의 허용되지 않는 경우 표시할 경고 메시지
stop	Function	호출되면 bAllowed 값과 상관 없이 모든 동작을 수행하지 않는다.

**onload**

업로드가 성공했음을 의미한다. 커스텀 이벤트 핸들링 예제는 다음과 같다

```
oComponent.attach("success", function(oCustomEvent) { ... });
```

onload 객체의 속성은 다음과 같다.

이름	타입	설명
sType	String	커스텀 이벤트 이름
htResult	HashTable	서버에서 전달해주는 결과 객체. 서버 설정에 따라 유동적으로 선택할 수 있다.

**onerror**

업로드가 실패했음을 의미한다. 커스텀 이벤트 핸들링 예제는 다음과 같다

```
oComponent.attach("error", function(oCustomEvent) { ... });
```

onerror 객체의 속성은 다음과 같다.

이름	타입	설명
sType	String	커스텀 이벤트 이름
htResult	HashTable	서버에서 전달해주는 결과 객체. 에러 발생 시 오류 메시지 문자열 값을 갖는 errMsgr 속성을 반드시 포함하도록 서버 응답을 설정해야 한다.

**메서드**

jindo.FileUploader의 메서드는 다음과 같다.

**표 12 jindo.FileUploader 메서드**

이름	설명	반환 값
getFileSelect	File Select 요소를 가져온다.	HTMLElement

이름	설명	반환 값
getFormElement	File Select 의 해당 Form 요소를 가져온다.	HTMLElement
reset	File Select 의 선택 값을 초기화한다.	
upload	IFRAME 으로 업로드를 수행한다.	