Installation

RELEASE 2.2.0 OR ABOVE

---

1. Installation

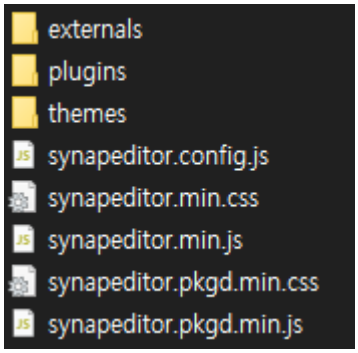
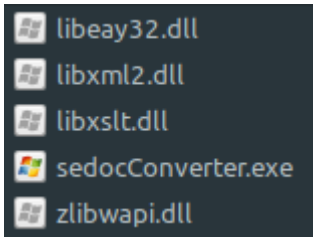
SynapEditor installation package is comprised of the following components.

- SynapEditor + Import Module
    - SynapEditor_2.x.x.zip
- Fonts for metafile conversion
    - fonts.zip

Copy the installation file to the desired path and decompress the zip file.

*In this Guide, we copy the file into /workspace and decompress the zip file. Explanations are based on the assumption that /workspace is located in WEB ROOT.

When you decompress the SynapEditor_2.x.x.zip file, the following path is generated.

Path	Content	Explanation
/workspace/SynapEditor_2.x.x /SynapEditor		externals : External module directory plugins : Plugin directory theme : theme directory license.json : License file synapeditor.config.js : Configuration file synapeditor.min.css, synapedtiro.min.js : SynapEditor synapeditor.pkgd.min.css, synapeditor.pkgd.min.js : SynapEditor including plugins
/workspace/SynapEditor_2.x.x /sedocConverter	Windows  Linux 	Import module (execution file) Windows : sedocConverter.exe and multiple dll files Linux : sedocConverter_exe

Decompress fonts.zip file under /workspace.

Path	Explanation
/workspace/fonts	Font file used to convert metafiles (wmf/emf) during document import

2. Configuration

2.1 License Configuration

Open the synapeditor.config.js file and designate the path for license.json file. The path for license.json file should be accessible via the browser.

synapeditor.config.js

```
{
   /**
     * Set up the path or object for the license file
     * ex) '/synapeditor/license.json'
     * ex)  {
                'company': 'Synapsoft',
                'key': [
                    'licenseKey'
                ]
            }
     */
    'editor.license': 'synapeditor/license.json',
  ...
}
```

3. 3. Initializing SynapEditor and Saving Edited Contents

Incorporating SynapEditor into your own environment is also very simple and straightforward.
You can use either or <textarea> tag for the instantiation of SynapEditor.

3.1 Using tag

3.1.1 Include js and css file for SynapEditor with <script> and <link> tags

```
<link href='synapeditor/synapeditor.min.css' rel='stylesheet' type='text/css'>
<script src='synapeditor/synapeditor.config.js'></script>
<script src='synapeditor/synapeditor.min.js'></script>
```

3.1.2 Create new SynapEditor instance using tag

```
<!DOCTYPE html>
<html lang="ko">

<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0,
user-scalable=no, shrink-to-fit=no">
<title>Synap Editor | Unlimited Rich Text Editor</title>

<link href="synapeditor/synapeditor.min.css" rel="stylesheet" type="text/css">
<script src="synapeditor/synapeditor.config.js"></script>
<script src="synapeditor/synapeditor.min.js"></script>
<script>
function initEditor() {
        var se = new SynapEditor("synapEditor", synapEditorConfig);
}
</script>
<body onload="initEditor();">
        <div id="synapEditor"></div>
</body>
</html>
```

3.1.3 Save edited content by POSTing HTML form

This example uses jQuery to send POST request.

```
<!DOCTYPE html>
<html lang="ko">

<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0,
user-scalable=no, shrink-to-fit=no">
<title>Synap Editor | Unlimited Rich Text Editor</title>
<link href="synapeditor/synapeditor.min.css" rel="stylesheet" type="text/css">
<script src="https://code.jquery.com/jquery-3.3.1.js"></script>
<script src="synapeditor/synapeditor.config.js"></script>
<script src="synapeditor/synapeditor.min.js"></script>
<script>
$(document).ready(function() {
        var se = new SynapEditor("synapEditor", synapEditorConfig);

        $('#seform').on('submit', function() {
                $('#editor').val( se.getPublishingHtml() );
                return true;
        });
});

</script>
<body>
        <div id="synapEditor"></div>
        <form id="seform" name="seform" action="/save" method="post">
                <textarea id="editor" style="display:none"></textarea>
                <input type="submit" value="SAVE"/>
        </form>
</body>
</html>
```

3.2 Using <textarea> tag

3.2.1 Include script and css file for SynapEditor with <script> and <link> tags

```
<link href='synapeditor/synapeditor.min.css' rel='stylesheet' type='text/css'>
<script src='synapeditor/synapeditor.config.js'></script>
<script src='synapeditor/synapeditor.min.js'></script>
```

3.2.2 Create new SynapEditor instance using <textarea> tag

```
<!DOCTYPE html>
<html lang="ko">

<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0,
user-scalable=no, shrink-to-fit=no">
<title>Synap Editor | Unlimited Rich Text Editor</title>

<link href="synapeditor/synapeditor.min.css" rel="stylesheet" type="text/css">
<script src="synapeditor/synapeditor.config.js"></script>
<script src="synapeditor/synapeditor.min.js"></script>
<script>
function initEditor() {
        var se = new SynapEditor("synapEditor", synapEditorConfig);
}
</script>
<body onload="initEditor();">
        <textarea id="synapEditor"></textarea>
</body>
</html>
```

3.2.3 Save edited content by POSTing HTML form

This example uses jQuery to send POST request.

```
<!DOCTYPE html>
<html lang="ko">

<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0,
user-scalable=no, shrink-to-fit=no">
<title>Synap Editor | Unlimited Rich Text Editor</title>
<link href="synapeditor/synapeditor.min.css" rel="stylesheet" type="text/css">
<script src="https://code.jquery.com/jquery-3.3.1.js"></script>
<script src="synapeditor/synapeditor.config.js"></script>
<script src="synapeditor/synapeditor.min.js"></script>
<script>
$(document).ready(function() {
        var se = new SynapEditor("synapEditor", synapEditorConfig);
});
</script>
<body>
        <form id="seform" name="seform" action="/save" method="post">
                <textarea id="synapEditor"></textarea>
                <input type="submit" value="SAVE"/>
        </form>
</body>
</html>
```

4. Applying Plugins and External Modules

You can add the various plugins provided by SynapEditor as follows.

Please refer to Plugin page for more detailed information.

Loading Plugins

```
<!-- Note) Plugin files shall be created under SynapEditor file.-->

<!-- SynapEditor File-->
<link rel="stylesheet" href="../synapeditor.min.css">
<script src="../synapeditor.min.js"></script>

<!-- Plugin File-->
<link rel="stylesheet" href="../plugins/webAccessibilityChecker/webAccessibilityChecker.min.css">
<script src="../plugins/webAccessibilityChecker/webAccessibilityChecker.min.js"></script>
```

Generating Plugin UI

```
<script>
        new SynapEditor('ID of HTML element to initialize the Editor', {
                //...
                'editor.toolbar': [
                        //...
                        'webAccessibilityChecker' //If you wish to expose the buttons provided by a plugin, use
the name provided by the plugin for the name of UI, just as the exsiting Editor configurations.
                        //...
                ]
                //...
        });
</script>
```

Applying External Modules

Apply external modules like Code Mirror etc. to enjoy more powerful editing features.

External Module	
CodeMirror	You can display the source code prettier and edit it conveniently.
formulaParser	Expand the function so that you can use the Excel function in the editor table.
SEDocModelParser	Expands functionality to import doc, docx, xls, xlsx documents.
SEShapeManager	The function is expanded to enable import and mark-in of a document containing figures. This module must be applied to use the shape editing plug-in.

Please refer to External Module page for more detailed information.

Example of applying plugins and external modules

```
<!DOCTYPE html>
<html lang="ko">

<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0,
user-scalable=no, shrink-to-fit=no">
<title>Synap Editor | Unlimited Rich Text Editor</title>

<!-- Synap Editor -->
<link href="synapeditor/synapeditor.min.css" rel="stylesheet" type="text/css">
<script src="synapeditor/synapeditor.config.js"></script>
<script src="synapeditor/synapeditor.min.js"></script>


<!-- Synap Editor Plugins -->
<!-- Shape Editor / Release 2.8.0 or above -->
<script src="synapeditor/plugins/shapeEditor/shapeEditor.min.js"></script>
<link rel="stylesheet" href="synapeditor/plugins/shapeEditor/shapeEditor.min.css">
<!-- Personal Data Protection -->
<script src="synapeditor/plugins/personalDataProtection/personalDataProtection.min.js"></script>
<link rel="stylesheet" href="synapeditor/plugins/personalDataProtection/personalDataProtection.min.css">
<!-- Special Character/Emoji -->
<script src="synapeditor/plugins/characterPicker/characterPicker.min.js"></script>
<link rel="stylesheet" href="synapeditor/plugins/characterPicker/characterPicker.min.css">
<!-- Photo Editor -->
<script src="synapeditor/plugins/tuiImageEditor/tuiImageEditor.min.js"></script>
<link rel="stylesheet" href="synapeditor/plugins/tuiImageEditor/tuiImageEditor.min.css">
<!-- Web Accessibility Checker -->
<script src="synapeditor/plugins/webAccessibilityChecker/webAccessibilityChecker.min.min.js"></script>
<link rel="stylesheet" href="synapeditor/plugins/webAccessibilityChecker/webAccessibilityChecker.min.min.css">
<!-- horizontalLine Extension -->
<script src="synapeditor/plugins/horizontalLineExtension/horizontalLineExtension.min.min.js"></script>
<link rel="stylesheet" href="synapeditor/plugins/webAccessibilityChecker/webAccessibilityChecker.min.min.css">
<!-- quote Extension -->
<script src="synapeditor/plugins/quoteExtension/quoteExtension.min.min.js"></script>
<link rel="stylesheet" href="synapeditor/plugins/quoteExtension/quoteExtension.min.min.css">

<!-- Synap Editor Externals -->
<script type="text/javascript" src='synapeditor/externals/formulaParser/formula-parser.min.js'></script>
<!-- Release 2.8.0 or above -->
<script type="text/javascript" src='synapeditor/externals/SEDocModelParser/SEDocModelParser.min.js'></script>
<!-- Release 2.8.0 or above -->
<script type="text/javascript" src='synapeditor/externals/SEShapeManager/SEShapeManager.min.js'></script>
<!-- CodeMirror -->
<script type="text/javascript" src='synapeditor/externals/codeMirror/codemirror.min.js'></script>
<script type="text/javascript" src="synapeditor/externals/codeMirror/xml.min.js"></script>
<link rel='stylesheet' href='synapeditor/externals/codeMirror/codemirror.min.css'>

<script>
function initEditor() {
        var se = new SynapEditor("synapEditor", synapEditorConfig);
}
</script>
<body onload="initEditor();">
        <div id="synapEditor"></div>
</body>
</html>
```

synapeditor.config.js

```
{
'editor.toolbar': [
    //...,
    'personalDataProtection',
    'specialCharacter', 'emoji',
    'tuiImageEditor',
    'WebAccessibilityChecker',
    //...
],
}
```

4. Import API and Upload API Configuration

Set the import and upload API path to the configuration file for file (image, video, ...) upload and document (doc, docx, xls, xlsx) import.

APIs(/importDoc, /uploadImage, /uploadVideo, /uploadFIle) required for Import and upload should be implemented in Back-end. Please refer to Server Connection Manual.

synapeditor.config.js

```
{
    'editor.import.api': '/importDoc',
    'editor.upload.image.api': '/uploadImage',
    'editor.upload.video.api': '/uploadVideo',
    'editor.upload.file.api': '/uploadFile',
   ...
}
```

Please refer to Configuration page for more detailed information regarding other settings including other menu and toolbar properties.