ASP.NET (C#)

1. , ,

.

.

C#

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;

// absolute path to WEB ROOT directory
private const string WEB_ROOT_ABS_PATH = @"C:\workspace\SynapEditor_C#\SynapEditor\wwwroot";

// absolute path to directory where the files are to be uploaded
private readonly string UPLOAD_DIR_ABS_PATH = Path.Combine(WEB_ROOT_ABS_PATH, "uploads");

/*
 * file upload API
 */
[HttpPost("api/uploadFile")]
public IActionResult UploadFile(IFormFile file)
{
    string uploadFileAbsPath = SaveFileToDisk(file).Result;
    string uploadFileRelPath = Path.GetRelativePath(UPLOAD_DIR_ABS_PATH, uploadFileAbsPath);

    return Ok(new {uploadPath = uploadFileRelPath});
}

/*
 * save data received from client-side to local filesystem
 */
private async Task<string> SaveFileToDisk(IFormFile file)
{
    string extension = Path.GetExtension(file.FileName);
    string uploadFileName = $"{Guid.NewGuid()}{extension}";
    string uploadFileAbsPath = Path.Combine(UPLOAD_DIR_ABS_PATH, uploadFileName);

    using (var fileStream = new FileStream(uploadFileAbsPath, FileMode.Create))
    {
        await file.CopyToAsync(fileStream);
    }

    return uploadFileAbsPath;
}
```

2. HWP, Word, Excel

.

.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using ICSharpCode.SharpZipLib.Zip;
using ICSharpCode.SharpZipLib.Zip.Compression.Streams;

// absolute path to WEB ROOT directory
private const string WEB_ROOT_ABS_PATH = @"C:\workspace\SynapEditor_C#\SynapEditor\wwwroot";

// absolute path to temporary workspace for decompressing documents
private readonly string WORK_DIR_ABS_PATH = Path.Combine(WEB_ROOT_ABS_PATH, "works");

// absolute path to directory where the files are to be uploaded
private readonly string UPLOAD_DIR_ABS_PATH = Path.Combine(WEB_ROOT_ABS_PATH, "uploads");

/*
 * Top-level import API
 */
[HttpPost("api/importDoc")]
public async Task<IActionResult> ImportDoc(IFormFile file)
{
    //1. save document to disk
    string uploadFileAbsPath = SaveFileToDisk(file).Result;

    //2. convert document into model data
    string importAbsPath = ExecuteConverter(uploadFileAbsPath).Result;
    string importRelPath = Path.GetRelativePath(WEB_ROOT_ABS_PATH, importAbsPath);

    //3. read in model data
    // v2.3.0   document.word.pb document.pb
    string pbFileAbsPath = Path.Combine(importAbsPath, "document.pb");
    int[] serializedData = ReadPbData(pbFileAbsPath);

        // 4. return serialized model data and its location
    return Ok(new {serializedData, importPath = importRelPath});
}

/*
 * save document to disk
 */
private async Task<string> SaveFileToDisk(IFormFile file)
{
    string extension = Path.GetExtension(file.FileName);
    string uploadFileName = $"{Guid.NewGuid()}{extension}";
    string uploadFileAbsPath = Path.Combine(UPLOAD_DIR_ABS_PATH, uploadFileName);

    using (var fileStream = new FileStream(uploadFileAbsPath, FileMode.Create))
    {
        await file.CopyToAsync(fileStream);
    }

    return uploadFileAbsPath;
}

/*
 * execute external document import module
 */
private async Task<string> ExecuteConverter(string docFileAbsPath)
{
        // absolute path to import module (the module dll should reside in the same path)
    const string CONVERTER_ABS_PATH = @"C:\workspace\SynapEditor_C#\sedocConverter.exe";

        // absolute path to font directory needed to convert metafiles
    const string FONTS_DIR_ABS_PATH = @"C:\workspace\SynapEditor_C#\fonts";

        // absolute path to temporary workspace
```

```
    const string TEMP_DIR_ABS_PATH = @"C:\workspace\SynapEditor_C#\tmp";

        // absolute path to the resulting zipped file
    string outputAbsPath = Path.Combine(WORK_DIR_ABS_PATH, Path.GetFileName(docFileAbsPath));

    ProcessStartInfo startInfo = new ProcessStartInfo();
    startInfo.FileName = CONVERTER_ABS_PATH;
    startInfo.Arguments = $"-f {FONTS_DIR_ABS_PATH} {docFileAbsPath} {outputAbsPath} {TEMP_DIR_ABS_PATH}";
    startInfo.CreateNoWindow = false;
    startInfo.UseShellExecute = false;
    startInfo.WindowStyle = ProcessWindowStyle.Hidden;

    using (Process exeProcess = Process.Start(startInfo))
    {
                // failure if conversion takes more than 30 sec.
        if (exeProcess.WaitForExit(30000))
        {
            return outputAbsPath;
        }

        exeProcess.Kill();
        return "";
    }
}

/*
 * serialize document model data in order to Synap Editor front-end module can consume it.
 * (Caution: importing will stop working if the body of this function is modified)
 */
private static int[] ReadPbData(string pbFilePath)
{
    List<int> pbData = new List<int>();
    using (FileStream fileStream = new FileStream(pbFilePath, FileMode.Open, FileAccess.Read))
    {
        fileStream.Seek(16, SeekOrigin.Begin);

        using (InflaterInputStream inflaterInputStream = new InflaterInputStream(fileStream))
        {
            int count;
            byte[] buffer = new byte[1024];

            while ((count = inflaterInputStream.Read(buffer, 0, buffer.Length)) > 0)
            {
                for (int i = 0; i < count; i++)
                {
                    pbData.Add(buffer[i] & 0xFF);
                }
            }
        }
    }

    return pbData.ToArray();
}
```

- Java Spring Framework
- Java Servlet
- ASP.NET (C#)
- ASP(Classic)
- PHP
- PHP4
- Django
- Ruby On Rails
- Wordpress plugin