using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text.RegularExpressions;

using System.Web;

/// <summary>

/// UploadHandler 的摘要说明

/// </summary>

public class UploadHandler : Handler

{

public UploadConfig UploadConfig { get; private set; }

public UploadResult Result { get; private set; }

public UploadHandler(HttpContext context, UploadConfig config)

: base(context)

{

this.UploadConfig = config;

this.Result = new UploadResult() { State = UploadState.Unknown };

}

public override void Process()

{

byte[] uploadFileBytes = null;

string uploadFileName = null;

if (UploadConfig.Base64)

{

uploadFileName = UploadConfig.Base64Filename;

uploadFileBytes = Convert.FromBase64String(Request[UploadConfig.UploadFieldName]);

}

else

{

var file = Request.Files[UploadConfig.UploadFieldName];

uploadFileName = file.FileName;

if (!CheckFileType(uploadFileName))

{

Result.State = UploadState.TypeNotAllow;

WriteResult();

return;

}

if (!CheckFileSize(file.ContentLength))

{

Result.State = UploadState.SizeLimitExceed;

WriteResult();

return;

}

uploadFileBytes = new byte[file.ContentLength];

try

{

file.InputStream.Read(uploadFileBytes, 0, file.ContentLength);

}

catch (Exception)

{

Result.State = UploadState.NetworkError;

WriteResult();

}

}

Result.OriginFileName = uploadFileName;

var savePath = PathFormatter.Format(uploadFileName, UploadConfig.PathFormat);

var localPath = Server.MapPath(savePath);

try

{

if (!Directory.Exists(Path.GetDirectoryName(localPath)))

{

Directory.CreateDirectory(Path.GetDirectoryName(localPath));

}

File.WriteAllBytes(localPath, uploadFileBytes);

Result.Url = savePath;

Result.State = UploadState.Success;

}

catch (Exception e)

{

Result.State = UploadState.FileAccessError;

Result.ErrorMessage = e.Message;

}

finally

{

WriteResult();

}

}

private void WriteResult()

{

this.WriteJson(new

{

state = GetStateMessage(Result.State),

url = Result.Url,

title = Result.OriginFileName,

original = Result.OriginFileName,

error = Result.ErrorMessage

});

}

private string GetStateMessage(UploadState state)

{

switch (state)

{

case UploadState.Success:

return "SUCCESS";

case UploadState.FileAccessError:

return "文件访问出错，请检查写入权限";

case UploadState.SizeLimitExceed:

return "文件大小超出服务器限制";

case UploadState.TypeNotAllow:

return "不允许的文件格式";

case UploadState.NetworkError:

return "网络错误";

}

return "未知错误";

}

private bool CheckFileType(string filename)

{

var fileExtension = Path.GetExtension(filename).ToLower();

return UploadConfig.AllowExtensions.Select(x => x.ToLower()).Contains(fileExtension);

}

private bool CheckFileSize(int size)

{

return size < UploadConfig.SizeLimit;

}

}

public class UploadConfig

{

/// <summary>

/// 文件命名规则

/// </summary>

public string PathFormat { get; set; }

/// <summary>

/// 上传表单域名称

/// </summary>

public string UploadFieldName { get; set; }

/// <summary>

/// 上传大小限制

/// </summary>

public int SizeLimit { get; set; }

/// <summary>

/// 上传允许的文件格式

/// </summary>

public string[] AllowExtensions { get; set; }

/// <summary>

/// 文件是否以 Base64 的形式上传

/// </summary>

public bool Base64 { get; set; }

/// <summary>

/// Base64 字符串所表示的文件名

/// </summary>

public string Base64Filename { get; set; }

}

public class UploadResult

{

public UploadState State { get; set; }

public string Url { get; set; }

public string OriginFileName { get; set; }

public string ErrorMessage { get; set; }

}

public enum UploadState

{

Success = 0,

SizeLimitExceed = -1,

TypeNotAllow = -2,

FileAccessError = -3,

NetworkError = -4,

Unknown = 1,

}