SinGooCMS.Utility说明文档

SinGooCMS.Utility是一个工具类集合，包括配置、文件、日期、数据、序列化、反射、图像处理、网络、缓存、Web相关、加解密、压缩、类扩展等工具类，几乎囊括了开发中的所有工具需求！支持netstandard2.1和net framework 4.6.1。

开源地地：<https://github.com/SinGooCMS/SinGooCMSUtility>

（注：以下代码可以在单元测试项目中找到）

1. HttpContext

.net framework的system.web自带有httpcontext.

但在.net core中必须注入才行。

public static class ServiceCollectionExtension 扩展类实现了注入httpcontext的功能，包括两个方法：

AddStaticHttpContext 和 UseStaticHttpContext

在Startup.cs中添加以下代码：

public void ConfigureServices(IServiceCollection services)

{

services.AddStaticHttpContext();

…

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

app.UseStaticHttpContext();

…

}

1. 扩展类

使用扩展类需要先引用命令空间：SinGooCMS.Utility.Extension

1. DataTableExtension

数据集DataTable和实体类之间的互相转换！

如：

//新建一个实体类

public class Student

{

public string UserName { get; set; }

public int Age { get; set; }

}

DataTable dt = new DataTable();

dt.Columns.Add("UserName", typeof(string));

dt.Columns.Add("Age", typeof(int));

DataRow dr1 = dt.NewRow();

dr1["UserName"] = "jsonlee";

dr1["Age"] = 18;

dt.Rows.Add(dr1);

DataRow dr2 = dt.NewRow();

dr2["UserName"] = "刘备";

dr2["Age"] = 20;

dt.Rows.Add(dr2);

var lst = dt.ToEntities<Student>();

1. DataTypeChangeExtension

类型转换，如 int val = “1”.ToInt();

1. DateExtension

给定日期，取日期的周一、周日、月初，月尾、年初、年尾以及格式化成字符串的相关操作。

如 ：

var now = DateTime.Now;

Console.WriteLine($"周一：{now.GetMonday()} 周日：{now.GetSunday()}");

Console.WriteLine($"月初：{now.GetMonthFirstDay()} 月尾：{now.GetMonthLastDay()}");

Console.WriteLine($"季初：{now.GetQuarterFirstDay()} 季尾：{now.GetQuarterLastDay()}");

Console.WriteLine($"年初：{now.GetYearFirstDay()} 年尾：{now.GetYearLastDay()}");

Console.WriteLine($"本年第几周：{now.WeekOfYear()}");

1. IConvertibleExtension

类型任意转换

如：

var b = "true";

bool flag = b.To<bool>();

1. IEnumerableExtension

可枚举类型增加foreach及去重操作

如：

lst.Foreach((p)=> Console.WriteLine(“元素值；”+p));

1. ImageExtension

图片处理类，当前对象必须是Image类型。

在后面的图片处理中详细介绍。

1. IOExtension

文件流的相关操作。

在后面的IO操作中详细介绍。

1. JsonserializeExtension

Json数据的序列化和反序列化操作。

//序列化

var user = new Student() { UserName="jsonlee",Age=18 };

var json = user.ToJson();

//反序列化

var user2 = json.JsonToObject<Student>();

//匿名类型序列化

var json = "{\"UserName\":\"jsonlee\",\"Age\":18}";

var model = new { UserName = "", Age = 0 };

var obj = json.JsonToAnonymousObject(model);

1. LinqExtension

待处理

1. ObjectExtension

对象的克隆

1. ReflectExtension

反射处理，后面详细介绍。

1. ServiceCollectionExtension

.net core HttpContext注入。

1. StreamExtension

文件流操作。

1. StringExtension

字符串扩展。

如：

var builder = new StringBuilder();

builder.AppendFormat("123转人民币：{0} \r\n",123.9m.ToRMB());

builder.AppendFormat("123456789转掩码：{0} \r\n", "123456789".Mask());

Console.WriteLine(builder.ToString());

var dict = new Dictionary<string, string>();

dict.Add("username","刘备");

dict.Add("age", "18");

Console.WriteLine("url:"+dict.ToUrlSearch());

var urlText = "username=%E5%88%98%E5%A4%87&age=18";

Console.WriteLine("username:" + urlText.ToUrlDictionary()["username"]);

1. ValidateExtension

验证扩展类，满足开发中常用的的验证。

如：

“”.IsNullOrEmpty() //true

"16826375@qq.com".IsEmail() //true

1. XmlserializeExtension

xml数据的序列化和反序列化操作。

//序列化

var user = new Student() { UserName = "jsonlee", Age = 18 };

var xml = user.ToXml();

Console.WriteLine("user to json:" + xml);

//反序列化

var user2 = xml.XmlToObject<Student>();

1. Web相关工具
2. WebUtils

查询QueryString和Form提交的参数，没有提供参数时，取默认值。

如

var queryStr= WebUtils.GetQueryString(“username”); //querystring get查询

var queryStr= WebUtils.GetFormString(“username”); //form post提交

var url=WebUtils.GetAbsoluteUri() //获取全路径

注意：.net core需要先注入HttpContext

1. CacheUtils

缓存工具，使用系统自带的缓存。

如：

DateTime dateTime = DateTime.Now;

var cache = new CacheUtils();

var cacheDT = DateTime.Now;

if (cache.ContainKey("time"))

cacheDT = cache.Get<DateTime>("time");

else

cache.Insert<DateTime>("time", dateTime, 3600);

builder.Append($"当前时间：{dateTime.ToFormatString()} \r\n");

builder.Append($"缓存时间：{cacheDT.ToFormatString()} \r\n");

1. CookieUtils

一般用户登录，验证成功后会写入用户标识，比如用户ID等。CookieUtils用于对浏览器cookie的操作管理，对于现在隐私权的越来越重视，部门浏览器对保存cookie很谨慎，.net core在配置中需要设置宽松策略。

如：

CookieUtils.SetCookie("username", "jsonlee");

builder.Append($"username cookie: {CookieUtils.GetCookie("username")} \r\n");

1. SessionUtils

Session和Cookie的概念差不多，但Session是保存在服务端的。在跨平台的时候，建立把Session保存在数据库中，或者使用JWT。

在.net core中需要在Startup中声明：

public void ConfigureServices(IServiceCollection services)

{

services.AddSession();

…

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

app.UseSession();

…

}

使用方法：

SessionUtils.SetSession("username", System.Web.HttpUtility.UrlEncode("刘备"));

builder.Append($"username session: {System.Web.HttpUtility.UrlDecode(SessionUtils.GetSession("username"))} \r\n");

1. 序列化

包括对Json和XML的序列化操作。

参考JsonserializeExtension、XmlserializeExtension

1. 配置

配置工具ConfigUtils。

对于.net framework是读取Web.config/App.config

对于.net core 是读取 appsettings.json

Web.config/App.config 配置：

<connectionStrings>

<add name="SQLConnSTR" providerName="SqlServer" connectionString="server=(local);database=SinGooCMS-v1.6;uid=sa;pwd=123" />

</connectionStrings>

<appSettings>

<add key="EnableCache" value="True" />

<!--The license context used-->

<add key="EPPlus:ExcelPackage.LicenseContext" value="NonCommercial" />

</appSettings>

Appsetting.json 配置：

"ConnectionStrings": {

"ProviderName": "Sqlite",

"SQLConnSTR": "Data Source=db/singootop.db",

"RedisExchangeHosts": "127.0.0.1:6379,allowadmin=true"

},

"AppSettings": {

//是否启用缓存 建议开启

"EnableCache": "True"

},

"EPPlus": {

"ExcelPackage": {

"LicenseContext": "NonCommercial"

}

}

读取方式：

//连接字符串

Console.WriteLine($"ProviderName：{ConfigUtils.ProviderName}\r\n");

Console.WriteLine($"SQLConnSTR：{ConfigUtils.DefConnStr}\r\n");

//appsetting

Console.WriteLine($"EnableCache：{ConfigUtils.GetAppSetting<string>("EnableCache")}\r\n");

1. 图片处理

在项目中经常会上传图片文件，我们可以对图片进行适当的处理，比如对大文件压缩尺寸，创建缩略图供快速浏览，创建水印图用于保护权利等。

如：

//文件转图像类型

var image = ImageUtils.ReadFileToImage(filePath);

var bitmap = ImageUtils.ReadFileBitmap(filePath);

var stream = FileUtils.ReadFileToStream(filePath);

Console.WriteLine("image宽高：" + image.Width + " x " + image.Height);

Console.WriteLine("bitmap宽高：" + bitmap.Width + " x " + bitmap.Height);

Console.WriteLine("stream-length：" + stream.Length);

//base64和图片互转

var base64Str = image.ToBase64();

Console.WriteLine("base64string:"+base64Str);

ImageUtils.Base64StrToImage(base64Str).Save(@"f:\123.png");

var image = ImageUtils.ReadFileToImage(filePath);

//转黑白图片

image.ToBWPic().Save(@"f:\黑白.png");

//调整光暗

image.LDPic(50).Save(@"f:\光暗.png");

//反色

image.RePic().Save(@"f:\反色.png");

//浮雕

image.Relief().Save(@"f:\浮雕.png");

//拉伸

image.ResizeImage(500, 300).Save(@"f:\拉伸.png");

//滤色

image.ColorFilter().Save(@"f:\滤色.png");

//马赛克

image.Mosaic(10).Save(@"f:\马赛克.png");

//上下翻转

image.UpDownRev().Save(@"f:\上下翻转.png");

//左右翻转

image.LeftRightRev().Save(@"f:\左右翻转.png");

//压缩图片

image.Compress(1000, 600).Save(@"f:\压缩.png");

//裁切

image.CutImage(new System.Drawing.Rectangle(0, 0, 300, 200)).Save(@"f:\裁切.png");

var image = ImageUtils.ReadFileToImage(filePath);

//缩略图

image.ThumbnailImage(300,150).Save(@"f:\缩略图.png");

ImageUtils.MakeThumbnail(filePath, 500, 450);

//水印

//文字水印

//ImageUtils.AddTextWatermark(filePath, "这是一个测试");

//图片水印

string watermarkPic = @"F:\jsonlee\study\test\logo.png";

ImageUtils.AddImageWatermark(filePath, watermarkPic);

注意：对于图片扩展类，对象必须是Image，如果是BitMap对象，需要转换成Image才能使用扩展类。缩略图和水印图片会存储在原文件同一目录下。

1. IO操作
2. 压缩

//压缩 支持压缩成 zip文件格式

string baseDir = System.Environment.CurrentDirectory; //当前程序的目录

string path = ZipUtils.Zip(baseDir);

//解压 支持 zip、rar、7z文件格式的解压

ZipUtils.UnZip(@"F:\jsonlee\W3Cschool-v2.1.0-win32-x64.zip", @"F:\ABC");

1. 共享文件

共享文件的操作和本地文件的一样的。但首先要连接，连接使用网络命令net use。

ShareFileUtils.DisconnectAll(); //断开所有连接

string path = @"\\192.168.1.120";

string uid = @"luxshare\11000890";

string pwd = @"abc";

var client = ShareFileUtils.Connect(path, uid, pwd);

if (client != null)

{

//var file = client.GetFile("KPI.xlsx"); //取文件

//client.DownFile(@"KPI.xlsx", "f:"); //下载

client.UpFile(@"f:\web.config", "ABC"); //上传

}

1. 二维码

创建二维码图片

var img = QRCodeUtils.GenerateQrCode("http://www.baidu.com", 30);

img.Save(@"f:\qrcode.png",System.Drawing.Imaging.ImageFormat.Png);

1. 验证码

var captcha = CaptchaUtils.Create();

captcha.CheckCodeImg.WriteToFile(@"f:\Captcha.png"); //验证码图片

Console.WriteLine("验证码字符串："+captcha.CheckCodeString); //验证码字符串

在web项目中可以这样输入验证码图片和记录验证码到cookie

public IActionResult CheckCodeImg()

{

var iCode = CaptchaUtils.Create();

CookieUtils.SetCookie("vcode", DEncryptUtils.DESEncode(iCode.CheckCodeString), 3600);

return File(iCode.CheckCodeImg, @"image/png");

}

1. 视频截图

VideoUtils，用于对.mp4等视频进行截图。需要首先下载视频工具ffmpeg.exe

使用方法：

VideoUtils.Init(“ffmpeg.exe”). Save(“视频文件”,”截图文件”);

1. 目录/文件操作

目录/文件的创建、读取、删除等操作

var lst = FileUtils.GetAllFiles(@"F:\jsonlee\study"); //递归读取目录下所有文件（包括子多层目录下的文件）

1. Excel操作

使用第三方工具EPPlus对Excel进行操作，不依赖微软的office组件。

EPPlus是一款开源框架，开源地址是：<https://github.com/EPPlusSoftware/EPPlus>

EPPlus使用前需要在配置中加上 NonCommercial 标志！请注意查看相关授权协议！

Web.config：

<appSettings>

<!--The license context used-->

<add key="EPPlus:ExcelPackage.LicenseContext" value="NonCommercial" />

</appSettings>

appSettings.json：

{

"EPPlus": {

"ExcelPackage": {

"LicenseContext": " NonCommercial "

}

}

}

1. 读取单元格的值

EPPlusUtils. ReadCell(“kpi.xlsx”,1,1); //读取第一个单元格的值

1. 读取sheet到DataTable

DataTable dt = EPPlusUtils.Read(“kpi.xlsx”);

1. DataTable导出到Excel

EPPlusUtils. Export(dt,”export.xlsx”);

1. 网络操作
2. FtpClient

//创建ftp客户端

// FtpClient ftp = FtpClient.GetClient(serverIP, ftpUserName,ftpUserPwd);

FtpClient ftp = FtpClient.GetClient("39.108.247.193", "fbac32bc", "fc156f7f66");

ftp.Download("/web/web.config", "f:\\web.config",true); //从服务器下载文件

var lst = ftp.GetFiles("/web/Upload/pdf/"); //读取目录下的所有文件信息

Console.WriteLine(lst.ToJson());

1. IPUtils

查看客户端IP，使用纯真数据库解析IP地址！

//查看客户端IP

Var ip = IPUtils.GetIP();

//定位IP

var ip = "218.95.66.68";

Console.WriteLine("IP定位："+new IPScanner().IPLocation(ip)); //江西省吉安市电信

注意：IP定位，需要在程序目录中有纯真数据库：\IPData\ qqwry.dat

1. NetWorkUtils

模拟http的Get方法和Post方法

//get

var getStr = NetWorkUtils.HttpGet("http://www.singoo.top");

Console.WriteLine("get返回数据："+getStr); //获取网页的源代码

//post 模拟用户登录

var postStr=NetWorkUtils.HttpPost("http://passports.singoo.top/passports/login", "\_loginname=admin&\_loginpwd=123");

Console.WriteLine("post返回数据：" + postStr); //post返回数据：登录失败，账号或者密码错误！

1. 加密解密

为了保护数据不被泄露，常常需要加密数据。数据加密分为可逆和不可逆两种方式。用户密码大多使用MD5加密，但原生的MD5加密，网络上有许多暴力破解，其实我们可以对数据进行2次加密或者多次加密，这样就不容易破解。

在API的数据往来传输中，同样需要加密，防止数据被截取导致信息泄露。比如支付宝、微信支付用的就是RSA加密，支付宝保存公钥，商户使用私钥。

1. 可逆加密解密

//可逆加密

//des

string desEncode = DEncryptUtils.DESEncrypt(txt, key);

Console.WriteLine("des加密值：" + desEncode);

Assert.AreEqual(txt, DEncryptUtils.DESDecrypt(desEncode, key));

//aes

string aesEncode = DEncryptUtils.AESEncrypt(txt);

Console.WriteLine("aes加密值：" + aesEncode);

Assert.AreEqual(txt, DEncryptUtils.AESDecrypt(aesEncode));

//base64

string base64Encode = DEncryptUtils.Base64Encrypt(txt);

Console.WriteLine("base64加密值：" + aesEncode);

Assert.AreEqual(txt, DEncryptUtils.Base64Decrypt(base64Encode));

//xor

string xor = DEncryptUtils.GetXORCode(txt, key);

Console.WriteLine("xor加密值：" + xor);

Assert.AreEqual(txt, DEncryptUtils.GetXORCode(xor, key));

1. 不可逆加密解密

//不可逆加密

//md5

string md5 = DEncryptUtils.MD5Encrypt(txt); //md5是2次加密，且会截断字符串，这样网上的暴力破解不了

Console.WriteLine("MD5:" + md5);

Assert.AreEqual(md5, DEncryptUtils.MD5Encrypt(txt));

//sha512 比md5更安全的不可逆加密方法

string sha512 = DEncryptUtils.SHA512Encrypt(txt);

Console.WriteLine("sha512:" + sha512);

Assert.AreEqual(sha512, DEncryptUtils.SHA512Encrypt(txt));

1. RSA加密解密

//RAS的公钥和私钥是分别保存的，保证了安全

//公钥

string publicKey = "";

//私钥

string privateKey = "";

RSACryption rsa = new RSACryption();

//生成公钥和私钥

rsa.RSAKey(out privateKey, out publicKey);

Console.WriteLine($"公钥：{publicKey} 私钥：{privateKey}");

//利用公钥加密

string encode = rsa.RSAEncrypt(publicKey, txt);

Console.WriteLine($"加密后的值：{encode}");

//利用私钥解密

string decode = rsa.RSADecrypt(privateKey, encode);

Console.WriteLine($"解密后的值：{decode}");

//判断

Assert.AreEqual(txt, decode);

十一、字符串处理

1. 简繁转换

//转为简体

var jian = StringUtils.GetSimplified(“今天天氣不錯，后天可能會下雪”);

//转为繁体

var fan = StringUtils.GetTraditional(“今天天气不错，后天可能会下雪”);

1. 随机数

var rnd = StringUtils.GetRandomString();

var filename = StringUtils.GetNewFileName();

1. 中文接音与首字母

string txtZHCN = "今天天气不错，后天可能会下雪";

Console.WriteLine($"【今天天气不错，后天可能会下雪】的拼音：{StringUtils.GetChineseSpell(txtZHCN)} \r\n");

Console.WriteLine($"【今天天气不错，后天可能会下雪】的拼音首字母：{StringUtils.GetChineseSpellFirst(txtZHCN)}\r\n");

十二、反射操作

//测试实体类

[Table("DbMaintenanceTest")]

public class DbMaintenanceTestInfo

{

private int id = 0;

public string uname = "jsonlee"; //private 读不到

[Key]

[NotMapped]

public int AutoID { get; set; }

public string UserName { get; set; }

}

反射可以通过dll文件，解析出里面的类、方法、属性、字段等，并调用这些对象。

readonly string connStr = "server=(local);database=TestDB;uid=sa;pwd=123;";

readonly string path = AppDomain.CurrentDomain.BaseDirectory + "SinGooCMS.Ado.dll";

readonly string className = "SinGooCMS.Ado.DbAccess.SqlServerAccess";

//反射获得对象

IDbAccess dbAccess = (IDbAccess)ReflectionUtil.CreateInstance(path, className, new object[] { connStr });

var model = dbAccess.Find<DbMaintenanceTestInfo>(1);

//反射获得对象

var instance = ReflectionUtil.CreateInstance(path, className, new object[] { connStr });

//反射调用泛型方法

var model = (DbMaintenanceTestInfo)instance.InvokeMethod("Find", new Type[] { typeof(DbMaintenanceTestInfo) }, new object[] { 1 });

var test = new DbMaintenanceTestInfo() { AutoID = 1, UserName = "jsonlee" };

//属性操作注意属性不能是私有的

var userName = test.GetProperty<string>("UserName");

test.SetProperty("UserName", "刘备");

//字段操作 注意字段不能是私有的

var fields = test.GetFields();

var val = test.GetField<string>("uname");

test.SetField("uname", "张飞");

十三、执行程序工具

//打开浏览器，并打网站

ProcessUtils.Execute(@"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe", "http://www.singoo.top");

//关机

ProcessUtils.Shutdown();

//执行cmd命令

ProcessUtils.ExecuteCMD(commandTexts);