Code Explanation Document

# using System;

引入系统命名空间

# using System.Net.Http;

引入用于HTTP请求的命名空间

# using System.Threading.Tasks;

引入支持异步编程的命名空间

# using System.Windows.Forms;

引入Windows窗体命名空间

# using Newtonsoft.Json;

引入Json处理库

# using System.IO;

引入文件输入输出命名空间

# namespace ReferenceGenerator

定义程序的主命名空间

# public partial class MainForm : Form

定义主窗体类，继承自Windows Forms的Form类

# private const string ApiKeyFilePath = "apikey.txt";

存储API密钥的文件路径

# private const string DefaultApiKey = "默认\_API\_KEY\_";

默认的API Key，但是我没有申请下来

# public MainForm()

构造函数，初始化窗体

# InitializeComponent();

初始化组件

# this.AutoScaleMode = AutoScaleMode.Dpi;

设置自动缩放模式为DPI，适应不同分辨率

# this.Resize += new EventHandler(this.MainForm\_Resize);

绑定窗体调整大小事件

# button1.Click += new EventHandler(this.Button1Click);

为按钮1点击事件绑定处理方法

# button2.Click += new EventHandler(this.Button2Click);

为按钮2点击事件绑定处理方法

# button3.Click += new EventHandler(this.Button3Click);

为按钮3点击事件绑定处理方法

# button4.Click += new EventHandler(this.Button4Click);

为按钮4点击事件绑定处理方法

# LoadApiKey();

加载API密钥

# private void MainForm\_Resize(object sender, EventArgs e)

窗体调整大小时的处理方法

# AdjustControlSizes();

调整控件大小和位置

# private void AdjustControlSizes()

调整控件大小和位置的方法

# listView1.Width = this.ClientSize.Width / 2;

调整ListView的宽度

# listView1.Height = this.ClientSize.Height - listView1.Top - 10;

调整ListView的高度

# richTextBox1.Left = listView1.Right + 10;

调整RichTextBox的左边距

# richTextBox1.Width = this.ClientSize.Width - richTextBox1.Left - 10;

调整RichTextBox的宽度

# richTextBox1.Height = this.ClientSize.Height - richTextBox1.Top - 10;

调整RichTextBox的高度

# private async void Button1Click(object sender, EventArgs e)

按钮1点击时的处理方法，异步执行

# string query = textBox1.Text;

获取用户输入的查询字符串

# string apiKey = textBox2.Text;

获取用户输入的API密钥

# if (string.IsNullOrWhiteSpace(query))

检查查询字符串是否为空

# MessageBox.Show("请输入正确的 DOI 或 paper 标题。");

提示用户输入正确的查询信息

# return;

终止操作

# if (string.IsNullOrWhiteSpace(apiKey))

检查API密钥是否为空

# MessageBox.Show("请确认你的 API Key 。");

提示用户输入API密钥

# try

尝试从API获取数据

# string response = await FetchDataFromApi(query, apiKey);

从API获取数据

# ParseAndDisplayResults(response);

解析并显示结果

# catch (HttpRequestException ex)

捕获HttpRequest异常

# MessageBox.Show(string.Format("网络错误: {0}", ex.Message));

提示网络错误信息

# catch (Exception ex)

捕获通用异常

# MessageBox.Show(string.Format("获取数据时发生错误: {0}", ex.Message));

提示获取数据时发生错误

# private async Task<string> FetchDataFromApi(string query, string apiKey)

从API获取数据的方法

# using (HttpClient client = new HttpClient())

创建一个HttpClient实例

# string url = string.Format("https://api.springer.com/metadata/json?q={0}&api\_key={1}", query, apiKey);

构建API的URL

# return await client.GetStringAsync(url);

发送HTTP GET请求，获取响应数据

# private void ParseAndDisplayResults(string jsonResponse)

解析JSON响应并更新ListView的方法

# dynamic results = JsonConvert.DeserializeObject(jsonResponse);

将JSON响应反序列化为动态对象

# listView1.Items.Clear();

清空ListView中的所有项

# foreach (var record in results.records)

遍历解析后的结果记录

# ListViewItem item = new ListViewItem(record.title.ToString());

创建新的ListViewItem

# item.Tag = record.bibtex.ToString();

将bibtex字符串存储在Tag属性中

# listView1.Items.Add(item);

添加项到ListView

# private void Button2Click(object sender, EventArgs e)

按钮2点击时的处理方法

# if (listView1.SelectedItems.Count > 0)

检查是否有选中的项

# string bibtex = listView1.SelectedItems[0].Tag.ToString();

获取选中项的bibtex信息

# string formattedReference = GenerateReferenceFromBibtex(bibtex);

生成格式化的参考文献

# richTextBox1.Text = formattedReference;

显示格式化的参考文献

# MessageBox.Show("请选择一个 bibtex 目录。");

提示用户选择一个bibtex目录

# private string GenerateReferenceFromBibtex(string bibtex)

从bibtex生成参考文献的方法

# return string.Format("格式化引用: {0}", bibtex);

返回格式化的引用字符串

# private void Button3Click(object sender, EventArgs e)

按钮3点击时的处理方法

# MessageBox.Show("请输入有效的 API Key 密钥");

提示用户输入有效的API密钥

# SaveApiKey(apiKey);

保存API密钥

# MessageBox.Show("API Key 密钥保存成功！");

提示保存成功

# private void Button4Click(object sender, EventArgs e)

按钮4点击时的处理方法

# string url = "https://dev.springernature.com/signup";

API注册页面的URL

# System.Diagnostics.Process.Start(url);

打开浏览器并访问URL

# MessageBox.Show(string.Format("打开网址时出错: {0}", ex.Message));

提示打开网址时发生错误

# private void SaveApiKey(string apiKey)

保存API密钥到文件的方法

# File.WriteAllText(ApiKeyFilePath, apiKey);

将API密钥写入文件

# private void LoadApiKey()

加载API密钥的方法

# if (File.Exists(ApiKeyFilePath))

检查文件是否存在

# string apiKey = File.ReadAllText(ApiKeyFilePath);

读取文件中的API密钥

# textBox2.Text = apiKey;

设置文本框的值

# textBox2.Text = DefaultApiKey;

如果文件不存在，使用默认的API密钥

# MessageBox.Show(string.Format("加载 API Key 密钥时出错: {0}", ex.Message));

提示加载密钥时发生错误