|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.04.05-01 51 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.04.05-01 12 01-1 | | **АГРЕГАТОР КУЛИНАРНЫХ РЕЦЕПТОВ НА ANDROID**  **СЕРВЕР**  **Текст программы**  **RU.17701729.04.05-01 12 01-1**  **Листов 43** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**Москва 2020**

ОГЛАВЛЕНИЕ

[1. ТЕКСТ ПРОГРАММЫ 4](#_Toc40315726)

[1.1. RecipeController.cs 4](#_Toc40315727)

[1.2. PageController.cs 6](#_Toc40315728)

[1.3. Recipe.cs 7](#_Toc40315729)

[1.4. RecipeContext 8](#_Toc40315730)

[1.5. appsettings.json 8](#_Toc40315731)

[1.6. nlog.config 9](#_Toc40315732)

[1.7. ServerMemoryCache.cs 10](#_Toc40315733)

[1.8. RecipeShort.cs 10](#_Toc40315734)

[1.9. RecipeFull.cs 11](#_Toc40315735)

[1.10. StepRecipe.cs 12](#_Toc40315736)

[1.11. Ingredient.cs 12](#_Toc40315737)

[1.12. Image.cs 13](#_Toc40315738)

[1.13. CPFC 13](#_Toc40315739)

[1.14. Additional.cs 14](#_Toc40315740)

[1.15. GetData.cs 15](#_Toc40315741)

[1.16. ParserException.cs 16](#_Toc40315742)

[1.17. HtmlLoaderPage.cs 17](#_Toc40315743)

[1.18. IParser.cs 18](#_Toc40315744)

[1.19. IParserPageSettings.cs 18](#_Toc40315745)

[1.20. ParserPage 19](#_Toc40315746)

[1.21. EdaPageParser.cs 20](#_Toc40315747)

[1.22. EdimDomaPageParser.cs 20](#_Toc40315748)

[1.23. PovarenokPageParser.cs 21](#_Toc40315749)

[1.24. PovarPageParser.cs 22](#_Toc40315750)

[1.25. EdaPageSettings.cs 22](#_Toc40315751)

[1.26. EdimDomaPageSettings.cs 23](#_Toc40315752)

[1.27. PovarenokPageSettings.cs 24](#_Toc40315753)

[1.28. PovarPageSettings.cs 25](#_Toc40315754)

[1.29. HtmlLoaderRecipe.cs 26](#_Toc40315755)

[1.30. IParserRecipe.cs 27](#_Toc40315756)

[1.31. IParserRecipeSettings.cs 27](#_Toc40315757)

[1.32. ParserRecipe.cs 28](#_Toc40315758)

[1.33. EdaRecipeParser.cs 28](#_Toc40315759)

[1.34. EdimDomaRecipeParser.cs 31](#_Toc40315760)

[1.35. PovarenokRecipeParser.cs 35](#_Toc40315761)

[1.36. PovarRecipeParser.cs 39](#_Toc40315762)

[1.37. EdaRecipeSettings.cs 42](#_Toc40315763)

[1.38. EdimDomaRecipeSettings.cs 42](#_Toc40315764)

[1.39. PovarenokRecipeSettings.cs 43](#_Toc40315765)

[1.40. PovarRecipeSettings.cs 43](#_Toc40315766)

# ТЕКСТ ПРОГРАММЫ

## RecipeController.cs

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using ObjectsLibrary;

using RecipeLibrary;

using System;

using System.Linq;

using System.Threading.Tasks;

using WebServer.Data;

namespace WebServer.Controllers

{

[ApiController]

// Переадресация на {ip adress}/recipe/

[Route("recipe")]

public class RecipeController : ControllerBase

{

private readonly ILogger<RecipeController> \_logger;

private const int diffHours = 24;

public RecipeController(ILogger<RecipeController> logger)

{

\_logger = logger;

}

/// <summary>

/// Метод, возвращающий по GET запросу полный кулинарный рецепт. <see cref="RecipeFull"/>

/// Маршрут: "{ip adress}/recipe/get?url={url}"

/// </summary>

/// <param name="url">URL адрес рецепта.</param>

/// <returns>Объект типа RecipeFull</returns>

[HttpGet("get")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public async Task<IActionResult> GetRecipe(string url)

{

\_logger.LogInformation($"Запрос на парсинг старницы рецепта. Url: {url}");

DateTime startTime = DateTime.Now;

RecipeFull recipe;

Recipe recipeDb;

using(RecipeContext db = new RecipeContext())

{

// Если в БД есть рецепт:

if((recipeDb = db.Recipes.FirstOrDefault(x => x.Url == url)) != null)

{

\_logger.LogDebug($"Рецепт существует в БД. Id = {recipeDb.Id}, Date = {recipeDb.Date}");

// Если рецепт нужно обновить:

if ((startTime - recipeDb.Date).TotalHours > diffHours)

{

\_logger.LogDebug($"Требуется обновление рецепта.");

try

{

recipe = await GetData.GetRecipe(url);

recipeDb.RecipeFull = db.RecipeToByte(recipe);

recipeDb.Date = startTime;

await db.SaveChangesAsync();

\_logger.LogInformation($"Статус: Ok.");

LogTime(startTime);

return Ok(recipe);

}

catch(Exception e)

{

\_logger.LogWarning(e, $"Произошла ошибка при парсинге рецепта! Выдан рецепт из БД.");

\_logger.LogInformation($"Статус: Ok.");

LogTime(startTime);

return Ok(db.ByteToRecipe(recipeDb.RecipeFull));

}

}

// Если рецепт есть в БД и обновляеть его ненужно:

else

{

\_logger.LogDebug("Обновление не требуется.");

\_logger.LogInformation($"Статус: Ok.");

LogTime(startTime);

recipe = db.ByteToRecipe(recipeDb.RecipeFull);

return Ok(recipe);

}

}

// Если в БД нет рецепта:

else

{

\_logger.LogDebug($"Рецепта не существует в БД.");

try

{

recipe = await GetData.GetRecipe(url);

await db.Recipes.AddAsync(new Recipe { Url = url, Date = startTime, RecipeFull = db.RecipeToByte(recipe) }) ;

await db.SaveChangesAsync();

\_logger.LogInformation($"Статус: Ok.");

LogTime(startTime);

return Ok(recipe);

}

catch (Exception e)

{

\_logger.LogWarning(e, $"Произошла ошибка при парсинге рецепта!");

\_logger.LogInformation($"Статус: 400");

LogTime(startTime);

return BadRequest();

}

}

}

}

private void LogTime(DateTime startTime)

{

\_logger.LogDebug($"Время исполнения: {(DateTime.Now - startTime).TotalMilliseconds} миллисекунд.");

}

}

}

## PageController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using ObjectsLibrary;

using RecipeLibrary;

using System;

using System.Threading.Tasks;

namespace WebServer.Controllers

{

[ApiController]

// Переадресация на {ip adress}/page/

[Route("page")]

public class PageController : ControllerBase

{

private readonly ILogger<PageController> \_logger;

public PageController(ILogger<PageController> logger)

{

\_logger = logger;

}

/// <summary>

/// Метод, возвращающий по GET запросу коллекцию кратких описаний кулинарных рецептов. <see cref="RecipeShort"/>

/// Маршрут: "{ip adress}/page/get?section={section}&page={page}&recipeName={recipeName}"

/// Параметры "page" и "recipeName" является необязательными.

/// </summary>

/// <param name="section">Раздел рецепта.</param>

/// <param name="page">Номер страницы.</param>

/// <param name="recipeName">Название рецепта.</param>

/// <returns>Объект типа RecipeShort[]</returns>

[HttpGet("get")]

public async Task<IActionResult> GetPages(string section, int page = 1, string recipeName = null)

{

string log = $"GET запрос на получение страниц с рецептами. Параметры: " +

$"section={section}, page={page}";

if (recipeName != null)

{

log += $", recipeName={recipeName}";

}

else

{

recipeName = string.Empty;

}

\_logger.LogInformation(log);

DateTime startTime = DateTime.Now;

try

{

RecipeShort[] recipes = await GetData.GetPage(section.ToLower(), page, recipeName.ToLower());

\_logger.LogDebug($"Время исполнения: {(DateTime.Now - startTime).TotalMilliseconds} миллисекунд.");

\_logger.LogInformation($"Статус: Ok.");

LogTime(startTime);

return Ok(recipes);

}

catch (Exception e)

{

\_logger.LogError(e, $"Запрос выполнен неудачно.");

\_logger.LogInformation($"Статус: 400.");

LogTime(startTime);

return BadRequest();

}

}

private void LogTime(DateTime startTime)

{

\_logger.LogDebug($"Время исполнения: {(DateTime.Now - startTime).TotalMilliseconds} миллисекунд.");

}

}

}

## Recipe.cs

using ObjectsLibrary;

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WebServer.Data

{

public class Recipe

{

[Key]

public Guid Id { get; set; }

public string Url { get; set; }

public DateTime Date { get; set; }

public byte[] RecipeFull { get; set; }

public Recipe()

{

//

}

}

}

## RecipeContext

using System.IO;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using NFX.Serialization.Slim;

using ObjectsLibrary;

namespace WebServer.Data

{

public class RecipeContext : DbContext

{

public DbSet<Recipe> Recipes { get; set; }

public RecipeContext()

{

Database.EnsureCreated();

}

public RecipeFull ByteToRecipe(byte[] arrBytes)

{

var memoryStream = new MemoryStream();

memoryStream.Write(arrBytes, 0, arrBytes.Length);

memoryStream.Seek(0, SeekOrigin.Begin);

SlimSerializer slimSerializer = new SlimSerializer();

return (RecipeFull) slimSerializer.Deserialize(memoryStream);

}

public byte[] RecipeToByte(RecipeFull recipeFull)

{

var memoryStream = new MemoryStream();

SlimSerializer slimSerializer = new SlimSerializer();

slimSerializer.Serialize(memoryStream, recipeFull);

return memoryStream.ToArray();

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseMySql("server=localhost;UserId=root;Password=password;database=recipes;");

}

}

}

## appsettings.json

{

"Database" :{

"Host": "localhost",

"Port": "3306",

"User": "root",

"Password": "password",

"Database": "recipes"

},

"Logging": {

"IncludeScopes": false,

"LogLevel": {

"Default": "Trace",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

## nlog.config

<?xml version="1.0" encoding="utf-8" ?>

<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

autoReload="true"

internalLogLevel="Info"

internalLogFile="internal-nlog.txt">

<extensions>

<add assembly="NLog.Web.AspNetCore"/>

</extensions>

<targets>

<target xsi:type="File" name="allfile" fileName="log/nlog-all-${shortdate}.log" layout="[${longdate}]: ${event-properties:item=EventId\_Id}|${uppercase:${level}}|${logger}|${message} ${exception:format=tostring}" />

<target xsi:type="File" name="ownFile-web" fileName="log/nlog-own-${shortdate}.log" layout="[${longdate}]: ${event-properties:item=EventId\_Id}|${uppercase:${level}}|${logger}|${message} ${exception:format=tostring}|url: ${aspnet-request-url}|action: ${aspnet-mvc-action}" />

<target name="logconsole" layout="${uppercase:${level}}|[${date}]: ${message}" xsi:type="ColoredConsole">

<highlight-word foregroundColor="DarkGreen" regex="^INFO" />

<highlight-word foregroundColor="Yellow" regex="^WARN" />

<highlight-word foregroundColor="Black" backgroundColor="Red" regex="^FAIL" />

<highlight-word foregroundColor="White" backgroundColor="Red" regex="^CRIT" />

</target>

<target name="logDebug" xsi:type="File" fileName="log/debug.log" layout="[${longdate}]: ${event-properties:item=EventId\_Id}|${uppercase:${level}}|${logger}|${message} ${exception:format=tostring}" />

<target name="logInfo" xsi:type="File" fileName="log/info.log" layout="[${longdate}]: ${event-properties:item=EventId\_Id}|${uppercase:${level}}|${logger}|${message} ${exception:format=tostring}" />

<target name="logWarning" xsi:type="File" fileName="log/warning.log" layout="[${longdate}]: ${event-properties:item=EventId\_Id}|${uppercase:${level}}|${logger}|${message} ${exception:format=tostring}" />

</targets>

<rules>

<logger name="Debugger" minlevel="Debug" writeTo="logDebug"/>

<logger minlevel="Info" writeTo="logInfo"/>

<logger minlevel="Warning" writeTo="logWarning"/>

<logger name="\*" minlevel="Trace" writeTo="allfile" />

<logger name="\*" minlevel="Info" writeTo="logconsole" />

<logger name="Microsoft.\*" maxlevel="Info" final="true" />

<logger name="\*" minlevel="Trace" writeTo="ownFile-web" />

</rules>

</nlog>

## ServerMemoryCache.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Extensions.Caching.Memory;

namespace WebServer

{

public class ServerMemoryCache

{

public MemoryCache Cache { get; }

public ServerMemoryCache()

{

Cache = new MemoryCache(new MemoryCacheOptions

{

SizeLimit = 1536

});

}

}

}

## RecipeShort.cs

using ObjectsLibrary.Components;

using System;

using System.Runtime.Serialization;

namespace ObjectsLibrary

{

/// <summary>Краткое описание рецепта, полученное со страницы поиска кулинарного сайта.</summary>

[Serializable]

public class RecipeShort

{

/// <value>Название рецепта.</value>

public string Title { get; set; }

/// <value>Изображение рецепта.</value>

/// /// <see cref="Image"/>ИК

public Image Image { get; set; }

/// <value>Интернет адрес рецепта.</value>

public string Url { get; set; }

/// <value>Индекс популярности рецепта.</value>

/// <remarks>Используется для сортировки рецептов сервером.</remarks>

[IgnoreDataMember]

public double IndexPopularity { get; set; }

public RecipeShort(string title, Image image, string url) : this()

{

Title = title;

Image = image;

Url = url;

}

public RecipeShort(string title, Image image, string url, double indexPopularity) : this(title, image, url)

{

IndexPopularity = indexPopularity;

}

public RecipeShort()

{

//

}

}

}

## RecipeFull.cs

using ObjectsLibrary.Components;

using System;

using System.ComponentModel.DataAnnotations;

using System.Runtime.Serialization;

namespace ObjectsLibrary

{

/// <summary>Полное описание рецепта.</summary>

[Serializable]

public class RecipeFull

{

/// <value>Адрес рецепта.</value>

public string Url { get; set; }

/// <value>Название рецепта.</value>

public string Title { get; set; }

/// <value>Главное изображение.</value>

/// <see cref="Image"/>

public Image TitleImage { get; set; }

/// <value>Описание рецепта.</value>

public string Description { get; set; }

/// <value>Массив ингредиентов рецепта.</value>

/// <see cref="Ingredient"/>

public Ingredient[] Ingredients { get; set; }

/// <value>Шаги приготовления рецепта.</value>

/// <see cref="StepRecipe"/>

public StepRecipe[] StepsRecipe { get; set; }

/// <value>Дополнительное описание рецепта.</value>

/// <see cref="Additional"/>

public Additional Additional { get; set; }

public RecipeFull()

{

// (0-.oo)

}

public RecipeFull(string url, string title, Image titleImage, string description, Ingredient[] ingredients,

StepRecipe[] stepsRecipe, Additional additional) : this()

{

Url = url;

Title = title;

TitleImage = titleImage;

Description = description;

Ingredients = ingredients;

StepsRecipe = stepsRecipe;

Additional = additional;

}

}

}

## StepRecipe.cs

using System;

namespace ObjectsLibrary.Components

{

/// <summary>Шаг приготовления кулинарного рецепта.</summary>

[Serializable]

public class StepRecipe

{

/// <value>Описание шага кулинарного рецепта.</value>

public string Description { get; set; }

/// <value>Изображение шага кулинарного рецепта.</value>

/// <see cref="Image"/>

public Image Image { get; set; }

public StepRecipe()

{

//

}

public StepRecipe(string description, Image image) : this()

{

Description = description;

Image = image;

}

}

}

## Ingredient.cs

using System;

namespace ObjectsLibrary.Components

{

/// <summary>Ингредиент.</summary>

[Serializable]

public class Ingredient

{

/// <value>Название ингредиента.</value>

public string Name { get; set; }

/// <value>Количество и мера измерения.</value>

public string Unit { get; set; }

/// <value>Название рецепта, которому принадлежит ингредиент.</value>

public string RecipeName { get; set; }

public Ingredient()

{

}

public Ingredient(string name, string unit, string recipeName) : this()

{

Name = name;

Unit = unit;

RecipeName = recipeName;

}

}

}

## Image.cs

using System;

namespace ObjectsLibrary.Components

{

/// <summary>Изображение.</summary>

[Serializable]

public class Image

{

/// <value>Интернет адрес на изображение.</value>

public string ImageUrl { get; set; }

public Image()

{

}

public Image(string url) : this()

{

ImageUrl = url;

}

}

}

## CPFC

using System;

namespace ObjectsLibrary.Components

{

/// <summary>Калории, белки, жиры, углеводы.</summary>

[Serializable]

public class CPFC

{

/// <value>Количество калорий.</value>

public double Calories { get; set; }

/// <value>Количество белка.</value>

public double Proteins { get; set; }

/// <value>Количество жиров.</value>

public double Fats { get; set; }

/// <value>Количество углеводов.</value>

public double Carbohydrates { get; set; }

public CPFC()

{

//

}

public CPFC(double calories, double proteins, double fats, double carbohydrates) : this()

{

Calories = calories;

Proteins = proteins;

Fats = fats;

Carbohydrates = carbohydrates;

}

}

}

## Additional.cs

using System;

namespace ObjectsLibrary.Components

{

/// <summary>Дополнительная информация рецепта.</summary>

[Serializable]

public class Additional

{

/// <value>Имя автора.</value>

public string AuthorName { get; set; }

/// <value>Количество порций.</value>

public int CountPortions { get; set; }

/// <value>Количество минут для приготовления блюда.</value>

public double PrepMinutes { get; set; }

/// <see cref="CPFC"/>

public CPFC CPFC { get; set; }

/// <value>Интернет адрес на видео.</value>

public string VideoUrl { get; set; }

public Additional()

{

//

}

public Additional(string authorName, int countPortions, double prepMinutes, CPFC cpfc) : this()

{

AuthorName = authorName;

CountPortions = countPortions;

PrepMinutes = prepMinutes;

CPFC = cpfc;

}

public Additional(string authorName, int countPortions, double prepMinutes,CPFC cpfc, string videoUrl) :

this(authorName, countPortions, prepMinutes, cpfc)

{

VideoUrl = videoUrl;

}

}

}

## GetData.cs

using ObjectsLibrary;

using ObjectsLibrary.Parser;

using ObjectsLibrary.Parser.ParserPage.Core;

using ObjectsLibrary.Parser.ParserPage.WebSites;

using ObjectsLibrary.Parser.ParserRecipe.Core;

using ObjectsLibrary.Parser.ParserRecipe.WebSites;

using RecipeLibrary.Parser.ParserPage.WebSites;

using RecipeLibrary.Parser.ParserRecipe.WebSites;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace RecipeLibrary

{

public class GetData

{

public static async Task<RecipeShort[]> GetPage(string section, int page, string findName = null)

{

var recipeShorts = new List<RecipeShort>();

var povarenok = new ParserPage<RecipeShort[]>

(new PovarenokPageParser(), new PovarenokPageSettings(section, page, findName));

var povar = new ParserPage<RecipeShort[]>

(new PovarPageParser(), new PovarPageSettings(section, page, findName));

var edimdoma = new ParserPage<RecipeShort[]>

(new EdimDomaPageParser(), new EdimDomaPageSettings(section, page, findName));

var eda = new ParserPage<RecipeShort[]>

(new EdaPageParser(), new EdaPageSettings(section, page, findName));

await Task.WhenAll(ParseRecipe(edimdoma, recipeShorts), ParseRecipe(povarenok, recipeShorts),

ParseRecipe(povar, recipeShorts), ParseRecipe(eda, recipeShorts));

return recipeShorts.OrderByDescending(x => x.IndexPopularity).ToArray();

}

private static async Task ParseRecipe(ParserPage<RecipeShort[]> T, List<RecipeShort> recipeShorts)

=> recipeShorts.AddRange(await T.Worker());

public static async Task<RecipeFull> GetRecipe(string url)

{

IParserRecipeSettings settings;

IParserRecipe<RecipeFull> obj;

if (url.Contains("www.povarenok.ru"))

{

obj = new PovarenokRecipeParser();

settings = new PovarenokRecipeSettings(url);

}

else if (url.Contains("povar.ru"))

{

obj = new PovarRecipeParser();

settings = new PovarRecipeSettings(url);

}

else if (url.Contains("www.edimdoma.ru"))

{

obj = new EdimDomaRecipeParser();

settings = new EdimDomaRecipeSettings(url);

}

else if (url.Contains("https://eda.ru"))

{

obj = new EdaRecipeParser();

settings = new EdaRecipeSettings(url);

}

else

throw new ParserException("Неизвестный сайт.");

var recipe = new ParserRecipe<RecipeFull>(obj, settings);

return await recipe.Worker();

}

}

}

## ParserException.cs

using System;

namespace ObjectsLibrary.Parser

{

[Serializable]

public class ParserException : System.Net.WebException

{

public ParserException()

{

//

}

public ParserException(string message) : base(message)

{

}

public ParserException(string message, Exception inner) : base(message, inner)

{

//

}

protected ParserException(System.Runtime.Serialization.SerializationInfo info,

System.Runtime.Serialization.StreamingContext context)

: base(info, context)

{

//

}

}

}

## HtmlLoaderPage.cs

using Newtonsoft.Json.Linq;

using System;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

namespace ObjectsLibrary.Parser.ParserPage.Core

{

public class HtmlLoaderPage

{

private readonly HttpClient \_client;

private readonly string \_url;

private readonly IParserPageSettings \_settings;

public HtmlLoaderPage(IParserPageSettings settings)

{

\_settings = settings;

\_url = settings.Url;

\_client = new HttpClient();

}

public async Task<string> GetSource(int idPage, string recipeName)

{

Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

if (!\_settings.Sections.ContainsKey(\_settings.Section))

{

throw new ParserException($"Раздела '{\_settings.Section}' не существует.");

}

string currentUrl = \_url + \_settings.Sections[\_settings.Section];

currentUrl = currentUrl

.Replace("{PageId}", idPage.ToString())

.Replace("{RecipeName}", recipeName);

HttpResponseMessage response;

// Для сайта eda.ru выполняются POST запросы, для остальных GET.

if (currentUrl.Contains("https://eda.ru"))

{

var data = "name=onthestove";

StringContent queryString = new StringContent(data);

response = await \_client.PostAsync(new Uri(currentUrl), queryString);

string responseBody = string.Empty;

if (response != null && response.StatusCode == HttpStatusCode.OK)

responseBody = await response.Content.ReadAsStringAsync();

var jObject = JObject.Parse(responseBody);

return jObject.ContainsKey("Recipes") ?

jObject["Recipes"]?.Value<string>() : jObject["Html"].Value<string>();

}

response = await \_client.GetAsync(currentUrl);

string source;

if (response != null && response.StatusCode == HttpStatusCode.OK)

source = await response.Content.ReadAsStringAsync();

else

throw new ParserException("Произошла ошибка при загрузке сайта");

return source;

}

}

}

## IParser.cs

using AngleSharp.Html.Dom;

namespace ObjectsLibrary.Parser.ParserPage.Core

{

internal interface IParserPage<out T> where T : class

{

T Parse(IHtmlDocument document, IParserPageSettings settings);

}

}

## IParserPageSettings.cs

using System.Collections.Generic;

namespace ObjectsLibrary.Parser.ParserPage.Core

{

public interface IParserPageSettings

{

string Url { get; }

string Section { get; }

Dictionary<string, string> Sections { get; }

int MaxPageId { get; }

int PageId { get; }

double IndexPopularity { get; set; }

double IndexStep { get; }

string RecipeName { get; }

}

}

## ParserPage

using AngleSharp.Html.Parser;

using System;

using System.Threading.Tasks;

namespace ObjectsLibrary.Parser.ParserPage.Core

{

internal class ParserPage<T> where T : class

{

private IParserPageSettings \_parserSettings;

private HtmlLoaderPage \_loader;

private static readonly Random Random = new Random();

private IParserPage<T> Parser { get; }

private IParserPageSettings Settings

{

get => \_parserSettings;

set

{

\_parserSettings = value;

\_loader = new HtmlLoaderPage(value);

}

}

private ParserPage(IParserPage<T> parser)

{

Parser = parser;

}

internal ParserPage(IParserPage<T> parser, IParserPageSettings parserSettings) : this(parser)

{

Settings = parserSettings;

}

private static int GetPageId(int maxPage) => Random.Next(1, maxPage + 1);

internal async Task<T> Worker()

{

var pageId = Settings.PageId;

if (Settings.Section == "random")

pageId = GetPageId(Settings.MaxPageId);

string recipeName = Settings.RecipeName;

string source = await \_loader.GetSource(pageId, recipeName);

// Если нет существуеющего раздела у сайта - выкидывается пустой.

if (source == null)

return null;

HtmlParser domParser = new HtmlParser();

var document = await domParser.ParseDocumentAsync(source);

var result = Parser.Parse(document, \_parserSettings);

return result;

}

}

}

## EdaPageParser.cs

using AngleSharp.Html.Dom;

using ObjectsLibrary.Parser.ParserPage.Core;

using System.Collections.Generic;

namespace ObjectsLibrary.Parser.ParserPage.WebSites

{

internal class EdaPageParser : IParserPage<RecipeShort[]>

{

public RecipeShort[] Parse(IHtmlDocument document, IParserPageSettings settings)

{

//tile-list\_\_horizontal-tile.horizontal-tile.js-portions-count-parent.js-bookmark\_\_obj

var recipes =

document.QuerySelectorAll(

"div.tile-list\_\_horizontal-tile.horizontal-tile.js-portions-count-parent.js-bookmark\_\_obj");

List<RecipeShort> recipeShorts = new List<RecipeShort>(recipes.Length);

double index = settings.IndexPopularity + settings.IndexStep;

foreach (var recipe in recipes)

{

index -= settings.IndexStep;

var divInfo = recipe.QuerySelector("div.lazy-load-container");

if (divInfo != null)

{

string imageUrl = divInfo.Attributes[3]?.Value ?? "";

string title = divInfo.Attributes[1]?.Value ?? "";

string url = settings.Url + recipe.QuerySelector("div.horizontal-tile\_\_item-link.js-click-link")

?.Attributes[1]?.Value ?? "";

if (imageUrl != "" && title != "")

recipeShorts.Add(new RecipeShort(title, new ObjectsLibrary.Components.Image(imageUrl), url,

index));

}

}

return recipeShorts.ToArray();

}

}

}

## EdimDomaPageParser.cs

using AngleSharp.Html.Dom;

using ObjectsLibrary.Components;

using ObjectsLibrary.Parser.ParserPage.Core;

using System.Linq;

namespace ObjectsLibrary.Parser.ParserPage.WebSites

{

internal class EdimDomaPageParser : IParserPage<RecipeShort[]>

{

public RecipeShort[] Parse(IHtmlDocument document, IParserPageSettings settings)

{

var recipeCards = document.QuerySelectorAll("article")

.Where(x => x.ClassName != null && x.ClassName == "card")

.Select(x => x.FirstElementChild);

double indexStartPopularity = settings.IndexPopularity;

return (from recipeCard in recipeCards

let url = "https://www.edimdoma.ru/" + recipeCard.Attributes[0].Value

let title = recipeCard.FirstElementChild.FirstElementChild.Attributes[1].Value

let pictureUrl = recipeCard.FirstElementChild.FirstElementChild.Attributes[2].Value

let image = new Image(pictureUrl)

let indexPopularity = indexStartPopularity -= settings.IndexStep

select new RecipeShort(title, image, url, indexStartPopularity)).ToArray();

}

}

}

## PovarenokPageParser.cs

using AngleSharp.Html.Dom;

using ObjectsLibrary.Components;

using ObjectsLibrary.Parser.ParserPage.Core;

using System.Linq;

namespace ObjectsLibrary.Parser.ParserPage.WebSites

{

internal class PovarenokPageParser : IParserPage<RecipeShort[]>

{

public RecipeShort[] Parse(IHtmlDocument document, IParserPageSettings settings)

{

var recipesList = document.QuerySelectorAll("article").Where(x=> x.ClassName == "item-bl");

double indexStartPopularity = settings.IndexPopularity;

return (from recipeBlock in recipesList

let url = recipeBlock.QuerySelector("div.m-img.desktop-img.conima")

.FirstElementChild.Attributes[0].Value

let image = new Image(recipeBlock.QuerySelector("div.m-img.desktop-img.conima")

.FirstElementChild.FirstElementChild.Attributes[0].Value)

let title = recipeBlock.QuerySelector("h2").QuerySelector("a").TextContent

let indexPopularity = indexStartPopularity -= settings.IndexStep

select new RecipeShort(title, image, url, indexStartPopularity)).ToArray();

}

}

}

## PovarPageParser.cs

using AngleSharp.Html.Dom;

using ObjectsLibrary;

using ObjectsLibrary.Components;

using ObjectsLibrary.Parser.ParserPage.Core;

using System.Linq;

namespace RecipeLibrary.Parser.ParserPage.WebSites

{

internal class PovarPageParser : IParserPage<RecipeShort[]>

{

public RecipeShort[] Parse(IHtmlDocument document, IParserPageSettings settings)

{

var recipesBody = document.QuerySelectorAll("div.recipe\_list > div.recipe");

double indexStartPopularity = settings.IndexPopularity;

return (from recipe in recipesBody

let anyBody = recipe.QuerySelector("h3 > a")

let title = anyBody.TextContent

let url = "https://povar.ru" + anyBody.Attributes[0].Value

let pictureBody = recipe.QuerySelector("img")

let imageUrl = pictureBody.Attributes[0].Value

let indexPopularity = indexStartPopularity -= settings.IndexStep

select new RecipeShort(title, new Image(imageUrl), url, indexPopularity)).ToArray();

}

}

}

## EdaPageSettings.cs

using System.Collections.Generic;

using ObjectsLibrary.Parser.ParserPage.Core;

namespace ObjectsLibrary.Parser.ParserPage.WebSites

{

internal class EdaPageSettings : IParserPageSettings

{

public string Url { get; } = "https://eda.ru/";

public string Section { get; }

public Dictionary<string, string> Sections { get; } = new Dictionary<string, string>()

{

{"new", "RecipesCatalog/GetPage?sorting=date&page={PageId}"},

{"random", "RecipesCatalog/GetPage?sorting=rate&page={PageId}"},

{"popular", "RecipesCatalog/GetPage?sorting=rating&page={PageId}"},

{"recipe", "RecipesSearch/GetNextRecipes?sorting=rating&page={PageId}&q={RecipeName}&OnlyEdaChecked=false"},

{"горячее", "RecipesCatalog/GetPage?sorting=rate&page={PageId}&tags=osnovnye-blyuda"},

{"супы", "RecipesCatalog/GetPage?sorting=rate&page={PageId}&tags=supy"},

{"салаты", "RecipesCatalog/GetPage?sorting=rate&page={PageId}&tags=salaty"},

{"закуски", "RecipesCatalog/GetPage?sorting=rate&page={PageId}&tags=zakuski"},

{"выпечка", "RecipesCatalog/GetPage?sorting=rate&page={PageId}&tags=vypechka-deserty"},

{"десерты", "RecipesCatalog/GetPage?sorting=rate&page={PageId}&tags=vypechka-deserty"},

{"соусы", "RecipesCatalog/GetPage?sorting=rate&page={PageId}&tags=sousy-marinady"}

};

public int MaxPageId { get; } = 350;

public int PageId { get; }

public string RecipeName { get; }

public double IndexPopularity { get; set; } = 100;

public double IndexStep { get; } = 1;

private EdaPageSettings(string section, int pageId)

{

Section = section;

PageId = pageId;

}

public EdaPageSettings(string section, int pageId, string recipeName) : this(section, pageId)

{

RecipeName = recipeName.Replace(' ', '+');

}

}

}

## EdimDomaPageSettings.cs

using System.Collections.Generic;

using ObjectsLibrary.Parser.ParserPage.Core;

namespace ObjectsLibrary.Parser.ParserPage.WebSites

{

internal class EdimDomaPageSettings : IParserPageSettings

{

public string Url { get; } = "https://www.edimdoma.ru/";

public string Section { get; }

public Dictionary<string, string> Sections { get; } = new Dictionary<string, string>()

{

{"new", "retsepty?page={PageId}"},

{"random", "retsepty?page={PageId}"},

{"popular", "retsepty?with\_ingredient=&without\_ingredient=&user\_ids=&page={PageId}&field=popular&direction=desc"},

{"recipe", "retsepty?with\_ingredient=&without\_ingredient=&user\_ids=&page={PageId}&field=&direction=&query={RecipeName}"},

{"горячее", "retsepty?page={PageId}&tags[recipe\_category][]=основные+блюда"},

{"супы", "retsepty?page={PageId}&tags[recipe\_category][]=супы+и+бульоны"},

{"салаты", "retsepty?page={PageId}&tags[recipe\_category][]=салаты+и+винегреты"},

{"закуски", "retsepty?page={PageId}&tags[recipe\_category][]=закуски"},

{"выпечка", "retsepty?page={PageId}&tags[recipe\_category][]=выпечка"},

{"десерты", "retsepty?page={PageId}&tags[recipe\_category][]=десерты"},

{"соусы", "retsepty?page={PageId}&tags[recipe\_category][]=соусы+и+заправки"}

};

public int MaxPageId { get; } = 4672;

public int PageId { get; set; }

public string RecipeName { get; }

public double IndexPopularity { get; set; } = 100;

public double IndexStep { get; } = 1;

public EdimDomaPageSettings(string section, int pageId)

{

Section = section;

PageId = pageId;

}

public EdimDomaPageSettings(string section, int pageId, string recipeName) : this(section, pageId)

{

RecipeName = recipeName.Replace(' ', '+');

}

}

}

## PovarenokPageSettings.cs

using ObjectsLibrary.Parser.ParserPage.Core;

using System.Collections.Generic;

using System.Linq;

namespace ObjectsLibrary.Parser.ParserPage.WebSites

{

internal class PovarenokPageSettings : IParserPageSettings

{

public string Url { get; } = "https://www.povarenok.ru/recipes/";

public Dictionary<string, string> Sections { get; } = new Dictionary<string, string>()

{

{"new", "~{PageId}/?sort=new&order=desc"},

{"random", "~{PageId}/?sort=new&order=desc"},

{"popular", "~{PageId}/?sort=rating&order=desc"},

{"recipe", "search/~{PageId}/?name={RecipeName}#searchformtop"},

{"горячее", "category/6/~{PageId}/"},

{"супы", "category/2/~{PageId}/"},

{"салаты", "category/12/~{PageId}/"},

{"закуски", "category/15/~{PageId}/"},

{"выпечка", "category/25/~{PageId}/"},

{"десерты", "category/30/~{PageId}/"},

{"соусы", "category/23/~{PageId}/"}

};

public string Section { get; }

public int MaxPageId { get; } = 9234;

public int PageId { get; }

public string RecipeName { get; }

public double IndexPopularity { get; set; } = 100;

public double IndexStep { get; } = 2;

public PovarenokPageSettings(string section, int pageId)

{

Section = section;

PageId = pageId;

}

public PovarenokPageSettings(string section, int pageId, string recipeName) : this(section, pageId)

{

RecipeName = GetUrl(recipeName);

}

private static string GetUrl(string recipeName)

{

// абвгдеёжзийклмнопрстуфхцчшщъыьэюя =>

// %E0%E1%E2%E3%E4%E5%B8%E6%E7%E8%E9%EA%EB%EC%ED%EE%EF%F0%F1%F2%F3%F4%F5%F6%F7%F8%F9%FA%FB%FC%FD%FE%FF

var valuePairs = new Dictionary<char, string>()

{

{'а', "E0"}, {'б', "E1"}, {'в', "E2"}, {'г', "E3"}, {'д', "E4"}, {'е', "E5"},

{'ё', "B8"}, {'ж', "E6"}, {'з', "E7"}, {'и', "E8"}, {'й', "E9"}, {'к', "EA"},

{'л', "EB"}, {'м', "EC"}, {'н', "ED"}, {'о', "EE"}, {'п', "EF"}, {'р', "F0"},

{'с', "F1"}, {'т', "F2"}, {'у', "F3"}, {'ф', "F4"}, {'х', "F5"}, {'ц', "F6"},

{'ч', "F7"}, {'ш', "F8"}, {'щ', "F9"}, {'ъ', "FA"}, {'ы', "FB"}, {'ь', "FC"},

{'э', "FD"}, {'ю', "FE"}, {'я', "FF"}

};

string url = string.Empty;

foreach (char symb in recipeName)

{

if (valuePairs.ContainsKey(symb))

url += '%' + valuePairs[symb];

if (symb == ' ')

url += '+';

}

return url;

}

}

}

## PovarPageSettings.cs

using System.Collections.Generic;

using ObjectsLibrary.Parser.ParserPage.Core;

namespace ObjectsLibrary.Parser.ParserPage.WebSites

{

internal class PovarPageSettings : IParserPageSettings

{

public string Url { get; } = "https://povar.ru/";

public string Section { get; }

public Dictionary<string, string> Sections { get; } = new Dictionary<string, string>()

{

{"new", "mostnew/all/{PageId}/"},

{"random", "mostnew/all/{PageId}/"},

{"popular", "master/rating/all/{PageId}/"},

{"recipe", "xmlsearch?query={RecipeName}&page={PageId}"},

{"горячее", "master/goryachie\_bliuda/{PageId}/"},

{"супы", "master/soup/{PageId}/"},

{"салаты", "master/salad/{PageId}/"},

{"закуски", "master/zakuski/{PageId}/"},

{"выпечка", "master/vypechka/{PageId}/"},

{"десерты", "master/dessert/{PageId}/"},

{"соусы", "master/sousy/{PageId}/"}

};

public int MaxPageId { get; set; } = 1788;

public int PageId { get; set; }

public string RecipeName { get; }

public double IndexPopularity { get; set; } = 100;

public double IndexStep { get; } = 1;

public PovarPageSettings(string section, int pageId)

{

Section = section;

PageId = pageId;

}

public PovarPageSettings(string section, int pageId, string recipeName) : this(section, pageId)

{

RecipeName = recipeName;

}

}

}

## HtmlLoaderRecipe.cs

using System.Net;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

namespace ObjectsLibrary.Parser.ParserRecipe.Core

{

public class HtmlLoaderRecipe

{

private readonly HttpClient \_client;

public HtmlLoaderRecipe(IParserRecipeSettings settings)

{

\_client = new HttpClient();

}

internal async Task<string> GetSource(string url)

{

var currentUrl = url;

var response = await \_client.GetAsync(currentUrl);

string source;

Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

if (response != null && response.StatusCode == HttpStatusCode.OK)

source = await response.Content.ReadAsStringAsync();

else

throw new ParserException("Error loading page");

return source;

}

}

}

## IParserRecipe.cs

using AngleSharp.Html.Dom;

namespace ObjectsLibrary.Parser.ParserRecipe.Core

{

internal interface IParserRecipe<out T> where T : RecipeFull

{

/// <summary>

/// Преобразует исходный <see cref="IHtmlDocument"/> IHtmlDocument в объект типа RecipeFull.

/// </summary>

/// <param name="document">Исходная веб-страница.</param>

/// <param name="settings">Настройки для парсинга веб странички.</param>

/// <returns>Объект вида <see cref="RecipeFull"/> RecipeFull</returns>

T Parse(IHtmlDocument document, IParserRecipeSettings settings);

/// <summary>

/// Преобразует входную строку и извлекает из неё количество минут,

/// требуемое для приготовления блюда.

/// Например: 10 часов 5 минут, 32 мин, 1 ч. и 10 мин и т.д

/// </summary>

/// <param name="inputLine">Входная строка, содержащия время приготовлени кулинарного рецепта.</param>

/// <returns>Количество минут.</returns>

double ConvertToMinutes(string inputLine);

}

}

## IParserRecipeSettings.cs

namespace ObjectsLibrary.Parser.ParserRecipe.Core

{

public interface IParserRecipeSettings

{

string Url { get; }

}

}

## ParserRecipe.cs

using AngleSharp.Html.Parser;

using System;

using System.Threading.Tasks;

namespace ObjectsLibrary.Parser.ParserRecipe.Core

{

internal class ParserRecipe<T> where T : RecipeFull

{

private IParserRecipeSettings parserSettings;

private HtmlLoaderRecipe loader;

private IParserRecipe<T> Parser { get; }

private IParserRecipeSettings Settings

{

get => parserSettings;

set

{

parserSettings = value;

loader = new HtmlLoaderRecipe(value);

}

}

private ParserRecipe(IParserRecipe<T> parser)

{

Parser = parser;

}

public ParserRecipe(IParserRecipe<T> parser, IParserRecipeSettings parserSettings) : this(parser)

{

Settings = parserSettings;

}

internal async Task<T> Worker()

{

try

{

var source = await loader.GetSource(Settings.Url);

var domParser = new HtmlParser();

var document = await domParser.ParseDocumentAsync(source);

var result = Parser.Parse(document, parserSettings);

return result;

}

catch (Exception e)

{

throw new ParserException("Ошибка при парсинге страницы: " + e);

}

}

}

}

## EdaRecipeParser.cs

using AngleSharp.Html.Dom;

using Newtonsoft.Json.Linq;

using ObjectsLibrary.Components;

using ObjectsLibrary.Parser.ParserRecipe.Core;

using System;

using System.Collections.Generic;

using System.Linq;

namespace ObjectsLibrary.Parser.ParserRecipe.WebSites

{

public class EdaRecipeParser : IParserRecipe<RecipeFull>

{

/// <see cref="RecipeFull.Url"/>

private string Url { get; set; }

/// <see cref="RecipeFull.Title"/>

private string Title { get; set; }

/// <see cref="RecipeFull.TitleImage"/>

private Image TitleImage { get; set; }

/// <see cref="RecipeFull.Description"/>

private string Description { get; set; }

/// <see cref="RecipeFull.Ingredients"/>

private Ingredient[] Ingredients { get; set; }

/// <see cref="RecipeFull.StepsRecipe"/>

private StepRecipe[] StepsRecipe { get; set; }

/// <see cref="RecipeFull.Additional"/>

private Additional Additional { get; set; }

public RecipeFull Parse(IHtmlDocument document, IParserRecipeSettings parserRecipeSettings)

{

Url = parserRecipeSettings.Url;

Title = document.QuerySelector("h1.recipe\_\_name.g-h1").TextContent.Trim().Replace("\n", "");

var divsDescription = document.QuerySelectorAll("p.recipe\_\_description");

var divsHistory = document.QuerySelectorAll("div.recipe\_\_story");

foreach (var description in divsDescription)

{

Description += description.TextContent.Trim().Replace("\n", "") + Environment.NewLine;

}

foreach (var history in divsHistory)

{

Description += history.TextContent.Trim().Replace("\n", "").Replace("Читать полностью", "") +

Environment.NewLine;

}

var pIngredients = document.QuerySelectorAll(

"div.ingredients-list.layout\_\_content-col > div.ingredients-list\_\_content > p.ingredients-list\_\_content-item.content-item.js-cart-ingredients");

List<Ingredient> ingredients = new List<Ingredient>(pIngredients.Length);

foreach (var pIngredient in pIngredients)

{

JObject jObject = JObject.Parse(pIngredient.Attributes[1].Value);

string title = jObject["name"].Value<string>();

string unit = jObject["amount"].Value<string>();

ingredients.Add(new Ingredient(title, unit, Title));

}

Ingredients = ingredients.ToArray();

var liSteps = document.QuerySelectorAll("ul.recipe\_\_steps > li.instruction.clearfix.js-steps\_\_parent");

List<StepRecipe> stepRecipes = new List<StepRecipe>(liSteps.Length);

foreach (var liStep in liSteps)

{

string imageUrl = liStep.QuerySelector("div.lazy-load-container")?.Attributes[3]?.Value ?? "";

string description = liStep.QuerySelector("span.instruction\_\_description.js-steps\_\_description")

.TextContent.Trim();

stepRecipes.Add(new StepRecipe(description, new Image(imageUrl)));

}

StepsRecipe = stepRecipes.ToArray();

// Берём последнюю итоговую фотографию рецепта:

var imageBody = document.QuerySelector("div.g-print-visible").QuerySelector("img");

if (imageBody?.Attributes[0] != null)

TitleImage = new Image(imageBody.Attributes[0].Value);

else

TitleImage = stepRecipes[stepRecipes.Count - 1]?.Image ??

new Image("https://s2.eda.ru/StaticContent/All/w/29261930/assets/images/png/404-ingr.png");

double prepMinutes = ConvertToMinutes(document.QuerySelectorAll("span.info-pad\_\_item")

.FirstOrDefault(x => x.FirstElementChild?.ClassName == "timer")

?.QuerySelector("span.info-text")?.TextContent ?? "0");

int countPortions = int.Parse(document.QuerySelectorAll("span.info-pad\_\_item")

.FirstOrDefault(x => x.FirstElementChild?.ClassName == "portion")

?.QuerySelector("span.info-text.js-portions-count-print")?.TextContent

.Split(' ')[0] ?? "0");

string authorName = document.QuerySelector("p.author-name > span")?.TextContent ?? string.Empty;

var videoScr = document.QuerySelector("iframe[frameborder='0']");

string videoUrl = string.Empty;

if (videoScr != null)

videoUrl = videoScr.Attributes[3]?.Value ?? "";

Additional = new Additional(authorName, countPortions, prepMinutes,

new CPFC(), videoUrl);

var cpfcList = document.QuerySelectorAll("li > p.nutrition\_\_weight");

if (cpfcList.Length > 3)

{

double calories = double.Parse(cpfcList[0]?.TextContent?.Replace('.', ',') ?? "0");

double protein = double.Parse(cpfcList[1]?.TextContent?.Replace('.', ',') ?? "0");

double fats = double.Parse(cpfcList[2]?.TextContent?.Replace('.', ',') ?? "0");

double carbohydrates = double.Parse(cpfcList[3]?.TextContent?.Replace('.', ',') ?? "0");

Additional = new Additional(authorName, countPortions, prepMinutes,

new CPFC(calories, protein, fats, carbohydrates), videoUrl);

}

return new RecipeFull(Url, Title, TitleImage, Description, Ingredients, StepsRecipe, Additional);

}

public double ConvertToMinutes(string inputLine)

{

if (inputLine is null)

return 0;

inputLine = inputLine.Replace(" и", String.Empty);

string[] arrayWords = inputLine.Split(' ');

double minutes = 0;

for (int i = 1; i < arrayWords.Length; i += 2)

{

if (arrayWords[i].Contains('м'))

minutes += int.Parse(arrayWords[i - 1]);

else if (arrayWords[i].Contains('ч'))

minutes += int.Parse(arrayWords[i - 1]) \* 60;

else if (arrayWords[i].Contains('д'))

// 60 \* 24 = 1440‬

minutes += int.Parse(arrayWords[i - 1]) \* 1440;

}

return minutes;

}

}

}

## EdimDomaRecipeParser.cs

using AngleSharp.Html.Dom;

using ObjectsLibrary.Components;

using ObjectsLibrary.Parser.ParserRecipe.Core;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text.RegularExpressions;

namespace ObjectsLibrary.Parser.ParserRecipe.WebSites

{

public class EdimDomaRecipeParser : IParserRecipe<RecipeFull>

{

/// <see cref="RecipeFull.Url"/>

private string Url { get; set; }

/// <see cref="RecipeFull.Title"/>

private string Title { get; set; }

/// <see cref="RecipeFull.TitleImage"/>

private Image TitleImage { get; set; }

/// <see cref="RecipeFull.Description"/>

private string Description { get; set; }

/// <see cref="RecipeFull.Ingredients"/>

private Ingredient[] Ingredients { get; set; }

/// <see cref="RecipeFull.StepsRecipe"/>

private StepRecipe[] StepsRecipe { get; set; }

/// <see cref="RecipeFull.Additional"/>

private Additional Additional { get; set; }

public RecipeFull Parse(IHtmlDocument document, IParserRecipeSettings parserRecipeSettings)

{

Url = parserRecipeSettings.Url;

var recipeBody =

document.QuerySelector("div.grid-three-column\_\_column.grid-three-column\_\_column\_center.onthe\_data")

?? throw new ParserException("Не найдено главное тело рецепта.");

Title = recipeBody.Attributes[5].Value ?? string.Empty;

TitleImage = new Image(recipeBody.QuerySelector("img").Attributes[2].Value ?? string.Empty);

Description = recipeBody.QuerySelector("div.recipe\_description")?.TextContent ?? string.Empty;

var ingredientBody = recipeBody.QuerySelector("div[id='recipe\_ingredients\_block']");

var inputArray = ingredientBody?.QuerySelectorAll("input.checkbox\_\_input.recipe\_ingredient\_checkbox");

if (inputArray != null)

{

List<Ingredient> ingredients = new List<Ingredient>(inputArray.Length);

foreach (var input in inputArray)

{

string titleIngredient = input.Attributes[2].Value;

string name = input.Attributes[4].Value;

string unit = input.Attributes[1].Value + ' ' + input.Attributes[6].Value;

name = titleIngredient == "Основные" ? name : name + " (" + titleIngredient + ')';

ingredients.Add(new Ingredient(name, unit, Title));

}

Ingredients = ingredients.ToArray();

}

var stepsBody = recipeBody.QuerySelector("div.recipe\_steps") ??

throw new ParserException("Не найден блок с шагами.");

var recipeArray = stepsBody.QuerySelectorAll("div.content-box.recipe\_step");

if (recipeArray != null)

{

List<StepRecipe> stepsRecipe = new List<StepRecipe>(recipeArray.Length);

stepsRecipe.AddRange(from stepBlock in recipeArray

let stepImage = new Image(

"https://www.edimdoma.ru" + stepBlock.QuerySelector("img")?.Attributes[0]?.Value ??

"/assets/default/recipe\_steps/ed4\_thumb-2c862fbcf2e544709c77a80ead4a3f58cd9a80e6b65f0ad18839af30ec9a2a5a.png")

let stepDescription = stepBlock.QuerySelector("div.plain-text.recipe\_step\_text")?.TextContent

select new StepRecipe(stepDescription, stepImage));

StepsRecipe = stepsRecipe.ToArray();

}

string authorName = recipeBody.QuerySelector("div.person\_\_name").TextContent;

int.TryParse(

recipeBody.QuerySelector("div.field\_\_container")?.FirstElementChild?.Attributes[3].Value ?? "0",

out int countPortions);

double prepMinutes = ConvertToMinutes(recipeBody.QuerySelector("div.entry-stats\_\_value").TextContent);

var cpfcDiv = recipeBody.QuerySelector("div.nutritional-value\_\_leftside");

var cpfc = new CPFC();

if (cpfcDiv != null)

{

double.TryParse(cpfcDiv.QuerySelector("div.kkal-meter\_\_value")?.TextContent ?? "0",

out double calories);

var tablePFC = cpfcDiv.QuerySelectorAll("div.nutritional-value\_\_nutritional-list > table");

if (tablePFC != null)

{

double.TryParse(

tablePFC[0].QuerySelector("td.definition-list-table\_\_td.definition-list-table\_\_td\_value")

.TextContent.Replace(" г", string.Empty) ?? "0", out double protein);

double.TryParse(

tablePFC[1].QuerySelector("td.definition-list-table\_\_td.definition-list-table\_\_td\_value")

.TextContent.Replace(" г", string.Empty) ?? "0", out double fats);

double.TryParse(

tablePFC[2].QuerySelector("td.definition-list-table\_\_td.definition-list-table\_\_td\_value")

.TextContent.Replace(" г", string.Empty) ?? "0", out double carbohydrates);

cpfc = new CPFC(calories, protein, fats, carbohydrates);

Additional = new Additional(authorName, countPortions, prepMinutes,

cpfc);

}

else

Additional = new Additional(authorName, countPortions, prepMinutes, cpfc);

}

// Так как JS скрипт подгружает видео позже, чем мы схватываем HTML

// А мне не хочется тратить лишнее время, чтобы его ждать

// То информацию о видео забираем прямо из JS скрипта)))

var videoBody = document.QuerySelectorAll("script[type = 'text/javascript']")

.FirstOrDefault(x => x.OuterHtml.Contains("var player = new Playerjs"))?.InnerHtml ??

null;

if (videoBody != null && videoBody != string.Empty)

{

// ,[720]//vid.edimdoma.ru/data/video/0008/0913/80913-original.mp4?1468483768","id":80913},{"

// => //vid.edimdoma.ru/data/video/0008/0913/80913-original.mp4?1468483768"

string videoUrl = string.Empty;

Regex regex = new Regex(@"(?s)(?<=720]).\*?(?="",)");

Match match = regex.Match(videoBody);

if (match.Success)

videoUrl = match.Value;

Additional = new Additional(authorName, countPortions, prepMinutes, cpfc, videoUrl);

}

return new RecipeFull(Url, Title, TitleImage, Description, Ingredients, StepsRecipe, Additional);

}

public double ConvertToMinutes(string inputLine)

{

if (inputLine is null)

return 0;

inputLine = inputLine.Replace(" и", String.Empty);

string[] arrayWords = inputLine.Split(' ');

double minutes = 0;

for (int i = 1; i < arrayWords.Length; i += 2)

{

if (arrayWords[i].Contains('м'))

minutes += int.Parse(arrayWords[i - 1]);

else if (arrayWords[i].Contains('ч'))

minutes += int.Parse(arrayWords[i - 1]) \* 60;

else if (arrayWords[i].Contains('д'))

// 60 \* 24 = 1440‬

minutes += int.Parse(arrayWords[i - 1]) \* 1440;

}

return minutes;

}

}

}

## PovarenokRecipeParser.cs

using AngleSharp.Html.Dom;

using ObjectsLibrary;

using ObjectsLibrary.Components;

using ObjectsLibrary.Parser.ParserRecipe.Core;

using System;

using System.Collections.Generic;

using System.Linq;

using AngleSharp.Dom;

namespace RecipeLibrary.Parser.ParserRecipe.WebSites

{

public class PovarenokRecipeParser : IParserRecipe<RecipeFull>

{

/// <see cref="RecipeFull.Url"/>

private string Url { get; set; }

/// <see cref="RecipeFull.Title"/>

private string Title { get; set; }

/// <see cref="RecipeFull.TitleImage"/>

private Image TitleImage { get; set; }

/// <see cref="RecipeFull.Description"/>

private string Description { get; set; }

/// <see cref="RecipeFull.Ingredients"/>

private Ingredient[] Ingredients { get; set; }

/// <see cref="RecipeFull.StepsRecipe"/>

private StepRecipe[] StepsRecipe { get; set; }

/// <see cref="RecipeFull.Additional"/>

private Additional Additional { get; set; }

private const string WhiteSpaceBug = " ";

/// <see cref="IParserRecipe{T}.Parse(IHtmlDocument, IParserRecipeSettings)"/>

public RecipeFull Parse(IHtmlDocument document, IParserRecipeSettings parserRecipeSettings)

{

var recipeBody = document.QuerySelector("article.item-bl.item-about");

Url = parserRecipeSettings.Url;

Title = recipeBody.QuerySelector("h1").TextContent;

TitleImage = new Image(recipeBody.QuerySelector("img[itemprop='image']").Attributes[1]?.Value);

Description = recipeBody.QuerySelector("div.article-text")

.TextContent

.Replace("\n", String.Empty)

.Replace(" ", String.Empty);

var ingredientBody = recipeBody.QuerySelector("div.ingredients-bl");

int countIngredientTitles = ingredientBody.QuerySelectorAll("ul").Length;

var ingredientsList = new List<Ingredient>();

for (int i = 0; i < countIngredientTitles; i++)

{

// "Время приготовления" и "Количество порций".

int count = 0;

var p = ingredientBody?.QuerySelectorAll("p")

.Select(item => item.TextContent).ToArray();

if (p.Length != 0)

{

if (p.Any(element => element.Contains("Время приготовления:")))

count++;

if (p.Any(element => element.Contains("Количество порций:")))

count++;

}

string titleIngredient;

var titleBody = ingredientBody?.QuerySelectorAll("p").ToArray();

if (titleBody?.Length - count == 0)

titleIngredient = Title;

else

titleIngredient = ingredientBody?.QuerySelectorAll("p")

.Select(item => item.TextContent).ToArray()[i];

var ingredientsArray = ingredientBody?.QuerySelectorAll("ul")

.ToArray()[i]

.QuerySelectorAll("li")

.ToArray();

Ingredient[] ingredients = new Ingredient[ingredientsArray.Length];

for (int j = 0; j < ingredientsArray.Length; j++)

{

string name = ingredientsArray[j].QuerySelector("span[itemprop='name']")?.TextContent;

string unit = ingredientsArray[j].QuerySelector("span[itemprop='amount']")?.TextContent;

name += ingredientsArray[j].TextContent

.Replace(name ?? "old\_value", string.Empty)

.Replace(unit ?? "old\_value", string.Empty)

.Replace("\n", string.Empty)

.Replace(WhiteSpaceBug, string.Empty)

.Replace("—", string.Empty);

if (titleIngredient != Title)

name += $" ({titleIngredient})";

Ingredient ingredient = new Ingredient(name, unit, Title);

ingredients[j] = ingredient;

ingredientsList.AddRange(ingredients);

}

}

Ingredients = ingredientsList.ToArray();

var recipesArray = recipeBody.QuerySelectorAll("div.cooking-bl");

int countRecipes = recipesArray.Length;

List<StepRecipe> stepRecipeBoxes = new List<StepRecipe>(countRecipes);

for (int i = 0; i < countRecipes; i++)

{

string imageUrl = recipesArray[i]?.FirstElementChild?.FirstElementChild?.Attributes[2]?.Value;

string description = recipesArray[i]?.LastElementChild?.FirstElementChild?.TextContent;

var image = new Image(imageUrl);

var stepRecipeBox = new StepRecipe(description, image);

stepRecipeBoxes.Add(stepRecipeBox);

}

if (stepRecipeBoxes.Count == 0)

{

var body = recipeBody.QuerySelector("div.article-tags")?.PreviousElementSibling ?? null;

if (body != null)

stepRecipeBoxes.Add(new StepRecipe(body?.TextContent ?? string.Empty,

new Image("https://www.povarenok.ru/i/new3/logo.png?v=1")));

}

StepsRecipe = stepRecipeBoxes.ToArray();

string authorName = recipeBody.QuerySelector("a[title='Профиль пользователя']")?.TextContent;

var ingredientBodyP = ingredientBody.QuerySelectorAll("p");

double prepMinutes = ConvertToMinutes(ingredientBodyP

.Where(x => x.FirstElementChild.TextContent.Contains("Время приготовления"))

.Select(x => x.LastElementChild.TextContent).FirstOrDefault());

int countPortions = int.Parse(ingredientBodyP

.Where(

x => x.FirstElementChild.TextContent.Contains("Количество порций:"))

.Select(x => x.TextContent).FirstOrDefault()

?.Replace("Количество порций:", string.Empty) ?? "0");

var cpfc = new CPFC();

Additional = new Additional(authorName, countPortions, prepMinutes, cpfc);

var tableBody = recipeBody.QuerySelector("div[id='nae-value-bl']");

if (tableBody != null)

{

var tableCPFC = tableBody

.LastElementChild?

.FirstElementChild?

.QuerySelectorAll("tr")[3]

.QuerySelectorAll("strong")

.Select(x => x.TextContent)

.ToArray();

if (tableCPFC != null)

{

double calories = double.Parse(tableCPFC[0].Replace(" ккал", string.Empty).Replace('.', ','));

double protein = double.Parse(tableCPFC[1].Replace(" г", string.Empty).Replace('.', ','));

double fats = double.Parse(tableCPFC[2].Replace(" г", string.Empty).Replace('.', ','));

double carbohydrates = double.Parse(tableCPFC[3].Replace(" г", string.Empty).Replace('.', ','));

cpfc = new CPFC(calories, protein, fats, carbohydrates);

}

Additional = new Additional(authorName, countPortions, prepMinutes, cpfc);

}

// Видео контент при наличии:

var videoBody = recipeBody.QuerySelector("div.video-wrapper");

if (videoBody != null)

{

string videoUrl = videoBody.FirstElementChild?.Attributes[2]?.Value ?? string.Empty;

Additional = new Additional(authorName, countPortions, prepMinutes, cpfc, videoUrl);

}

return new RecipeFull(Url, Title, TitleImage, Description, Ingredients, StepsRecipe, Additional);

}

/// <see cref="IParserRecipe{T}.ConvertToMinutes(string)"/>

public double ConvertToMinutes(string inputLine)

{

if (inputLine == null)

return 0;

// 40 минут, 80 минут, 10 минут.

inputLine = inputLine.Replace(" и", String.Empty);

string[] arrayWords = inputLine.Split(' ');

double minutes = 0;

for (int i = 1; i < arrayWords.Length; i += 2)

{

if (arrayWords[i].Contains('м'))

minutes += int.Parse(arrayWords[i - 1]);

else if (arrayWords[i].Contains('ч'))

minutes += int.Parse(arrayWords[i - 1]) \* 60;

else if (arrayWords[i].Contains('д'))

// 60 \* 24 = 1440‬

minutes += int.Parse(arrayWords[i - 1]) \* 1440;

}

return minutes;

}

}

}

## PovarRecipeParser.cs

using AngleSharp.Html.Dom;

using ObjectsLibrary;

using ObjectsLibrary.Components;

using ObjectsLibrary.Parser.ParserRecipe.Core;

using System;

using System.Collections.Generic;

using System.Linq;

namespace RecipeLibrary.Parser.ParserRecipe.WebSites

{

public class PovarRecipeParser : IParserRecipe<RecipeFull>

{

/// <see cref="RecipeFull.Url"/>

private string Url { get; set; }

/// <see cref="RecipeFull.Title"/>

private string Title { get; set; }

/// <see cref="RecipeFull.TitleImage"/>

private Image TitleImage { get; set; }

/// <see cref="RecipeFull.Description"/>

private string Description { get; set; }

/// <see cref="RecipeFull.Ingredients"/>

private Ingredient[] Ingredients { get; set; }

/// <see cref="RecipeFull.StepsRecipe"/>

private StepRecipe[] StepsRecipe { get; set; }

/// <see cref="RecipeFull.Additional"/>

private Additional Additional { get; set; }

/// <see cref="IParserRecipe{T}.Parse(IHtmlDocument, IParserRecipeSettings)"/>

public RecipeFull Parse(IHtmlDocument document, IParserRecipeSettings parserRecipeSettings)

{

Url = parserRecipeSettings.Url;

var recipeBody = document.QuerySelector("div.cont\_area");

if (recipeBody is null)

return new RecipeFull();

Title = recipeBody.QuerySelector("h1.detailed[itemprop='name']").TextContent;

TitleImage = new Image(recipeBody.QuerySelector("div.bigImgBox")?

.QuerySelector("a").FirstElementChild.Attributes[0].Value);

Description = String.Empty;

foreach (var textLine in recipeBody.QuerySelectorAll("span.detailed\_full").Select(x => x.TextContent))

Description += textLine.Trim() + Environment.NewLine;

Description.Replace("\t", "").Replace(" ", "");

var ingredientsBody = recipeBody.QuerySelector("ul.detailed\_ingredients").QuerySelectorAll("li");

int countIngredients = ingredientsBody.Length;

Ingredient[] ingredients = new Ingredient[countIngredients];

for (int i = 0; i < countIngredients; i++)

{

string title = ingredientsBody[i].Attributes[1].Value;

string unit = ingredientsBody[i].TextContent

.Replace(title, string.Empty)

.Remove(0, 35);

ingredients[i] = new Ingredient(title, unit, Title);

}

Ingredients = ingredients;

var stepRecipeBody =

recipeBody.QuerySelector("div[itemprop='recipeInstructions'][itemtype='http://schema.org/ItemList']");

if (stepRecipeBody != null)

{

var stepCollection = stepRecipeBody.QuerySelectorAll("div");

List<StepRecipe> stepRecipesBoxes = new List<StepRecipe>(stepCollection.Length / 3);

foreach (var step in stepCollection)

{

switch (step.ClassName)

{

case "detailed\_step\_photo\_big":

{

var firstEl = step.FirstElementChild;

string description = firstEl.Attributes[0].Value;

string pictureUrl = firstEl.Attributes[3].Value;

stepRecipesBoxes.Add(new StepRecipe(description,

new Image(pictureUrl)));

break;

}

case "detailed\_step\_description\_big noPhotoStep":

{

string description = step.TextContent;

stepRecipesBoxes.Add(new StepRecipe(description,

new Image()));

break;

}

}

}

StepsRecipe = stepRecipesBoxes.ToArray();

}

var rcpAuthorTimeBody = recipeBody

.QuerySelector("div.rcpAuthorTime");

string authorName = rcpAuthorTimeBody.QuerySelector("span[id='autorName']").TextContent;

double prepMinutes =

ConvertToMinutes(rcpAuthorTimeBody.QuerySelector("meta[itemprop='cookTime']").Attributes[1].Value);

string[] portions = recipeBody

.QuerySelector("em[itemprop='recipeYield']")

.TextContent

.Remove(0, 19)

.Split('-') ?? null;

int countPortions = 0;

if (portions != null)

countPortions = int.Parse(RemoveSymbols(portions[0]));

// На повар.ру нет информации(

CPFC CPFC = null;

Additional = new Additional(authorName, countPortions, prepMinutes, CPFC);

var videoBody = recipeBody.QuerySelector("iframe.youtubeVideo");

if (videoBody != null)

{

string videoUrl = videoBody.Attributes[2]?.Value ?? string.Empty;

Additional = new Additional(authorName, countPortions, prepMinutes, CPFC, videoUrl);

}

return new RecipeFull(Url, Title, TitleImage, Description, Ingredients,

StepsRecipe,

Additional);

}

private static string RemoveSymbols(string line)

{

return line.Where(item => char.IsDigit(item)).Aggregate(string.Empty, (current, item) => current + item);

}

/// <see cref="IParserRecipe{T}.ConvertToMinutes(string)"/>

public double ConvertToMinutes(string inputLine)

{

// Format: PT<min>M

return double.Parse(inputLine[2..^1]);

}

}

}

## EdaRecipeSettings.cs

using ObjectsLibrary.Parser.ParserRecipe.Core;

namespace ObjectsLibrary.Parser.ParserRecipe.WebSites

{

public class EdaRecipeSettings : IParserRecipeSettings

{

public string Url { get; }

public EdaRecipeSettings(string url)

{

Url = url;

}

}

}

## EdimDomaRecipeSettings.cs

using ObjectsLibrary.Parser.ParserRecipe.Core;

namespace ObjectsLibrary.Parser.ParserRecipe.WebSites

{

public class EdimDomaRecipeSettings : IParserRecipeSettings

{

public string Url { get; }

public EdimDomaRecipeSettings(string url)

{

Url = url;

}

}

}

## PovarenokRecipeSettings.cs

using ObjectsLibrary.Parser.ParserRecipe.Core;

namespace ObjectsLibrary.Parser.ParserRecipe.WebSites

{

public class PovarenokRecipeSettings : IParserRecipeSettings

{

public string Url { get; }

public PovarenokRecipeSettings(string url)

{

Url = url;

}

}

}

## PovarRecipeSettings.cs

using ObjectsLibrary.Parser.ParserRecipe.Core;

namespace ObjectsLibrary.Parser.ParserRecipe.WebSites

{

public class PovarRecipeSettings : IParserRecipeSettings

{

public string Url { get; }

public PovarRecipeSettings(string url)

{

Url = url;

}

}

}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |