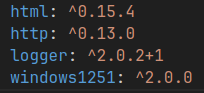
**Лабораторна робота № 3**

*Тема:* Введення в Flutter розробку під desktop

**Хід роботи**

Для початку створимо новий проект:  
flutter create webpage\_parser

Для парсингу сайту нам знадобляться такі пакети (можна додати за допомогою flutter pub add):



Потрібно визначитись з сайтом, який будемо парсити, та його адресою. Для цього створимо файл **data/constants/string\_constants.dart**  в папці **lib** та запишемо адресу сайта з завдання в константну змінну:

const hostname = "https://www.pisni.org.ua";

Можна починати роботу над сторінками. Створимо сторінку для категорій пісень. Її код буде виглядати наступним чином:  
import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:html/parser.dart' as html\_parser;

import 'package:logger/logger.dart';

import 'package:webpage\_parser/logic/decoded\_string.dart';

import 'package:webpage\_parser/presentation/pages/category\_songs\_page.dart';

import '../../data/constants/string\_constants.dart';

class CategoriesPage extends StatefulWidget {

const CategoriesPage({Key? key}) : super(key: key);

@override

State<CategoriesPage> createState() => \_CategoriesPageState();

}

class \_CategoriesPageState extends State<CategoriesPage> {

final String endpoint = '/songlist';

List<Map<String, String>> songCategories = [];

@override

void initState() {

super.initState();

fetchHTML();

}

void fetchHTML() async {

try {

final response = await http.get(Uri.parse(hostname + endpoint));

if (response.statusCode == 200) {

final document = html\_parser.parse(response.body);

final elements = document.getElementsByTagName('a');

for (final element in elements) {

String title = element.text.decoded;

if (title.isEmpty) {

continue;

}

final href = element.attributes['href'];

if (RegExp(r'^/songlist/.\*1\.html$').hasMatch(href ?? '')) {

final categoryMap = <String, String>{

'title': title,

'href': href ?? "",

};

songCategories.add(categoryMap);

final retrievedTitle = categoryMap['title'];

Logger().i('Retrieved Title: $retrievedTitle');

}

}

setState(() {});

} else {

Logger()

.e('Failed to fetch web page. Status code: ${response.statusCode}');

}

} catch (e) {

Logger().e('Error: $e');

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('Song Links'),

),

body: songCategories.isEmpty

? const Center(

child: CircularProgressIndicator(),

)

: ListView.builder(

itemCount: songCategories.length,

itemBuilder: (context, index) {

final categoryItem = songCategories[index];

return ListTile(

title: Text(categoryItem['title'] ?? 'title undefined'),

subtitle: Text(categoryItem['href'] ?? 'link undefined'),

onTap: () {

if (categoryItem['href'] != null) {

Navigator.of(context)

.push(MaterialPageRoute(builder: (context) {

return CategorySongsPage(link: categoryItem['href']!);

}));

return;

}

ScaffoldMessenger.of(context).clearSnackBars();

ScaffoldMessenger.of(context).showSnackBar(

const SnackBar(

content: Text("Sorry, your link is invalid"),

),

);

},

);

},

),

);

}

}

В класі стану цієї сторінки створено два поля: рядок з адресою сторінки, яка буде парситись та список з категоріями пісень. Це список буде заповнятись в методі **fetchHTML().**

Процес парсингу виглядає приблизно так:

1. Робимо **GET** запит на сторінку, який повертає всю html структуру сторінки:

final response = await http.get(Uri.parse(hostname + endpoint));

1. Далі, якщо запит вдалий, використовуючи **html/parser,** парсимо сторінку в клас Document, з яким далі і будемо працювати. Вибираємо з цього документа усі теги з ім’ям ‘**a**’:

final document = html\_parser.parse(response.body);

final elements = document.getElementsByTagName('a');

1. Після цього, в циклі обробляємо кожне вибране посилання, при цьому його валідуючи, і записуємо в список, де буде зберігатись вміст тегу та вміст **href** атрибута:

String title = element.text.decoded;

if (title.isEmpty) {

continue;

}

final href = element.attributes['href'];

if (RegExp(r'^/songlist/.\*1\.html$').hasMatch(href ?? '')) {

final categoryMap = <String, String>{

'title': title,

'href': href ?? "",

};

songCategories.add(categoryMap);

1. Коли список заповниться усіма посиланнями, виводимо його на сторінці, використовуючи **ListView.builder** віджет:

ListView.builder(

itemCount: songCategories.length,

itemBuilder: (context, index) {

final categoryItem = songCategories[index];

return ListTile(

title: Text(categoryItem['title'] ?? 'title undefined'),

subtitle: Text(categoryItem['href'] ?? 'link undefined'),

onTap: () {

if (categoryItem['href'] != null) {

Navigator.of(context)

.push(MaterialPageRoute(builder: (context) {

return CategorySongsPage(link: categoryItem['href']!);

}));

return;

}

ScaffoldMessenger.of(context).clearSnackBars();

ScaffoldMessenger.of(context).showSnackBar(

const SnackBar(

content: Text("Sorry, your link is invalid"),

),

);

},

);

},

)

Вкажемо цю сторінку як головну в файлі **main.dart**:

import 'package:flutter/material.dart';

import 'presentation/pages/categories\_page.dart';

void main() => runApp(

const MaterialApp(

debugShowCheckedModeBanner: false,

home: CategoriesPage(),

),

);

***Важливо!***

Варто звернути увагу на **meta** тег в шапці сторінки де записано властивість **charset**.

При отриманні даних через **http** потрібно ці дані розкодовувати за допомогою кодування з цього метатегу. Інакше дані будуть повертатись в не дуже коректному вигляді. В даному проекті використовується кодування **windows-1251***.* Для розкодовування даних з цим кодуванням використовується пакет з такою ж назвою.

Для швидкого розкодовування рядкових даних можна написати користувацьке доповнення для типу **String**. Воно виглядає наступним чином:

import 'package:windows1251/windows1251.dart';

extension DecodedString on String {

String get decoded => windows1251.decode(runes.toList());

}

Це працює таким чином, що в цьому проекті ми доповнили базовий клас типу **String** і тепер при роботі з рядковими даними в цьому проекті завжди буде доступний **getter** “**decoded**” , який буде повертати декодований рядок. Наприклад:

String title = element.text.decoded;

Властивість **element, text** є рядком, але з цього рядка можна викликати **decoded,** якого в стандартній реалізації рядка не існує.

Ця сторінка буде мати такий вигляд:



Аналогічним чином можна написати і інші сторінки. Наприклад сторінку, з піснями конкретної категорії **CategorySongsPage.dart**:

import 'package:flutter/material.dart';

import 'package:html/parser.dart' as html\_parser;

import 'package:http/http.dart' as http;

import 'package:logger/logger.dart';

import 'package:webpage\_parser/data/constants/string\_constants.dart';

import 'package:webpage\_parser/logic/decoded\_string.dart';

import 'package:webpage\_parser/presentation/pages/song\_page.dart';

class CategorySongsPage extends StatefulWidget {

const CategorySongsPage({super.key, required this.link});

final String link;

@override

State<CategorySongsPage> createState() => \_CategorySongsPageState();

}

class \_CategorySongsPageState extends State<CategorySongsPage> {

late String currentPageLink;

int currentPageIndex = 1;

late Future<List<Map<String, String>>> futureSongs;

late String currentPageTitle;

@override

void initState() {

currentPageLink = widget.link;

futureSongs = \_fetchSongs();

super.initState();

}

Future<List<Map<String, String>>> \_fetchSongs() async {

final url = Uri.parse(hostname + currentPageLink);

final response = await http.get(url);

if (response.statusCode == 200) {

final document = html\_parser.parse(response.body);

final pageTitle = document.getElementsByTagName("h1")[0].text.decoded;

currentPageTitle = pageTitle;

final tableRows = document.querySelectorAll('table.list tr.li');

final List<Map<String, String>> songs = [];

for (final row in tableRows) {

final titleElement = row.getElementsByTagName('a');

final tdElements = row.getElementsByTagName('td');

final title = titleElement[0].text.decoded;

final href = titleElement[0].attributes['href'].toString();

final authors = tdElements[1].text.decoded;

final songMap = {

'title': title,

'href': href,

'authors': authors,

};

songs.add(songMap);

}

return songs;

} else {

Logger().e('Failed to load HTML: ${response.statusCode}');

throw Exception('Failed to load HTML');

}

}

void \_loadNextPage() {

setState(() {

currentPageLink = currentPageLink.replaceAllMapped(

RegExp(r'\d+'),

(match) => (int.parse(match.group(0)!) + 1).toString(),

);

currentPageIndex += 1;

futureSongs = \_fetchSongs();

});

}

void \_loadPreviousPage() {

setState(() {

currentPageLink = currentPageLink.replaceAllMapped(

RegExp(r'\d+'),

(match) => (int.parse(match.group(0)!) - 1).toString(),

);

currentPageIndex -= 1;

futureSongs = \_fetchSongs();

});

}

@override

Widget build(BuildContext context) {

return FutureBuilder(

future: futureSongs,

builder: (context, snapshot) {

if (snapshot.connectionState == ConnectionState.waiting) {

return Scaffold(

appBar: AppBar(

title: const Text('Loading...'),

),

body: const Center(

child: CircularProgressIndicator(),

),

);

} else if (snapshot.hasError) {

return Scaffold(

appBar: AppBar(

title: const Text('Error'),

),

body: Center(

child: Text('Error: ${snapshot.error}'),

),

);

} else {

List<Map<String, String>> songs =

snapshot.data as List<Map<String, String>>;

return Scaffold(

appBar: AppBar(

title: Text(currentPageTitle),

),

body: Column(

children: [

Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

if (currentPageIndex != 1)

IconButton(

onPressed: \_loadPreviousPage,

icon: const Icon(Icons.chevron\_left),

iconSize: 32,

),

Text(

'Page: $currentPageIndex',

style: const TextStyle(fontSize: 22),

),

IconButton(

onPressed: \_loadNextPage,

icon: const Icon(Icons.chevron\_right),

iconSize: 32,

),

],

),

Expanded(

child: ListView.builder(

itemCount: songs.length,

itemBuilder: (context, index) {

final song = songs[index];

return ListTile(

title: Text(song['title'] ?? ''),

subtitle: Text(song['authors'] ?? ''),

trailing: Text(song['href'] ?? ''),

onTap: () {

if (song['href'] != null) {

Navigator.of(context).push(

MaterialPageRoute(

builder: (context) {

return SongPage(songLink: song['href']!);

},

),

);

return;

}

ScaffoldMessenger.of(context).clearSnackBars();

ScaffoldMessenger.of(context).showSnackBar(

const SnackBar(

content: Text("Sorry, your link is invalid"),

),

);

},

);

},

),

),

],

),

);

}

},

);

}

}

Ця сторінка приймає посилання на категорію як параметр в конструкторі віджета, а в класі стану вже проходить робота з цим посиланням(завантаження пісень, зміна сторінки і т.д.) Ця сторінка викликається за допомогою **Navigator.of(context).push()** методу на сторінці з категоріями при натиску на елемент категорії:

onTap: () {

if (categoryItem['href'] != null) {

Navigator.of(context)

.push(MaterialPageRoute(builder: (context) {

return CategorySongsPage(link: categoryItem['href']!);

}));

return;

}

ScaffoldMessenger.of(context).clearSnackBars();

ScaffoldMessenger.of(context).showSnackBar(

const SnackBar(

content: Text("Sorry, your link is invalid"),

),

);

},

Аналогічним чином можна написати сторінку для виведення даних про обрану пісню.

**Завдання**

Реалізувати програмне рішення з можливістю парсингу даних із сайту відповідно до варіанту завдання. Розробити зручний вивід даних.

*\* При можливості зробити вкладене завантаження, тобто якщо є посилання, то переходити по ньому і ще завантажувати дані з внутрішнього посилання.*

**Варіанти завдання:**

1. Спарсити заміни з сайту коледжу <https://hpk.edu.ua/replacements>
2. Спортивні новини з сайту <https://football.ua/>
3. Витягти інформацію про товари із будь-якого сайту.
4. Витягти інформацію про оголошення із будь-якого сайту.
5. Новини з сайту <https://24tv.ua/>
6. Інформацію з сайту <https://www.minusrus.com/>
7. Спортивні новини з будь-якого сайту.
8. Спарсити дані з сайту будь-якого новин.
9. Інформацію з сайту <https://russianwarship.rip/>
10. Класифікацію засобів з сайту <https://compendium.com.ua/uk/atc/>
11. Витягти новини із сайту <https://mil.in.ua/uk/>
12. Витягти інформацію про фільми із будь-якого сайту.
13. Інформацію про матч з сайту <https://football.ua/games_online/86555.html>
14. Витягти інформацію про автомобілі із будь-якого сайту.
15. Витягти інформацію про меблі із будь-якого сайту.
16. Витягти пісні із сайту <https://www.pisni.org.ua/songs/65858.html>
17. Витягти інформацію із сайту <https://dou.ua/>
18. Витягти музику із будь-якого сайту.
19. Витягти інформацію із сайту <https://uakino.club/>
20. Витягти інформацію із сайту [https://www.work.ua](https://www.work.ua/)

**Література**