Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет кібернетики

Звіт

Лабораторної роботи

Тема:

# ФАЙЛОВИЙ МЕНЕДЖЕР

Виконав студент 2 курсу

Романус Теодор Ігорович, К -24

Київ – 2015

# ЗМІСТ

[ФАЙЛОВИЙ МЕНЕДЖЕР 1](#_Toc438549731)

[ЗМІСТ 2](#_Toc438549732)

[Умова. 3](#_Toc438549733)

[Реалізація 3](#_Toc438549734)

[Елементи інтерфейсу: 4](#_Toc438549735)

[Код 4](#_Toc438549736)

[Реалізація методу DOM API 4](#_Toc438549737)

[Реалізація методу SAS API 5](#_Toc438549738)

[Реалізація методу LINQ to XML 7](#_Toc438549739)

[Перетворення XML в HTML 7](#_Toc438549740)

[Діаграма класів 9](#_Toc438549741)

[Список використаних джерел 10](#_Toc438549742)

Лабораторна робота 1-3

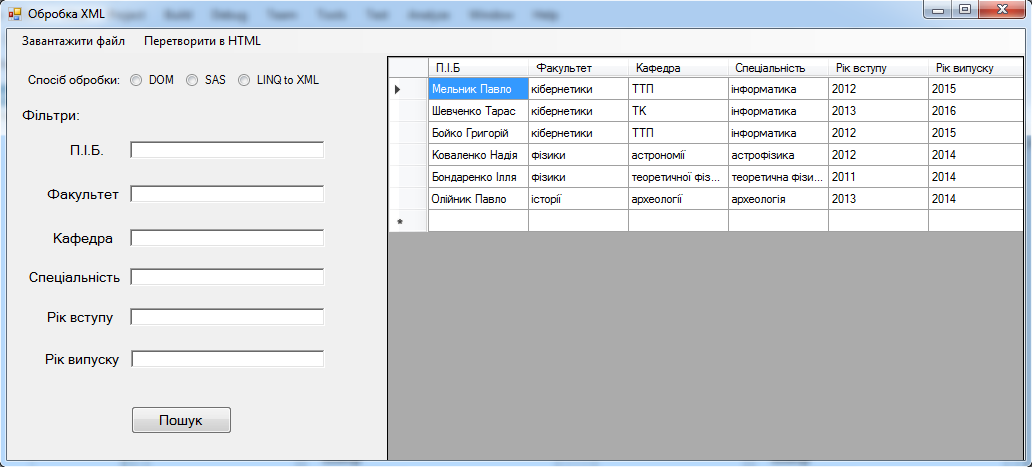
# Умова.

Реалізувати три способи обробки XML файлу - *SAX API*, *DOM API* та *LINQ to XML*. Обробка включає в себе дві задачі: аналіз вмісту документу (пошук інформації за ключовими словами та динамічну генерацію запитів) та трансформацію у файл HTML.

Індивідуальне завдання. "Облік випускників"

Таблиці містять таку інформацію: П.І.П., факультет (департамент, відділення), кафедра, спеціальність, час вступу та закінчення, переміщення з посади на посаду та з одного місця роботи на інше.

# Реалізація



Вікно роботи з XML запускається з файлового менеджера шляхом вибору відповідної функції в панелі меню.

Клас для роботи з XML – форма *XML Searcher*. Вікно включає в себе таблицю, поле для вибору методу фільтрації і поле для введення фільтрів.

Головні атрибути – шлях до файлу, над яким виконуватиметься обробка, дані в таблиці, збережені в форматі об’єкта *DataTable*, а також список усіх фільтрів в форматі *ListItem*.

Задля уникнення використання антипаттерну “*Magic Button*” уся логіка програми винесена в окремий модуль “*XML Implementation.cs*”.

В процесі розробки використовувався шаблон “*Strategy*”. Взаємодія усіх трьох фільтрів обробки з програмою відбувається через інтерфейс *IParser*.

# Елементи інтерфейсу:

1. Вибір файла здійснюється через діалогове вікно вибору файлу *OpenFileDialog* шляхом натискання кнопки «Завантажити файл» на панелі меню. Зауваження: Відкрити можна лише файл типу XML.
2. Введення фільтрів пошуку відбувається через поля-форми *RichTextBox.*
3. Власне пошук співпадінь по файлу здійснюється після натискання кнопки “Пошук”.
4. У процесі роботи допускається збереження таблиці у форматі *.html.* Щоб зберегти файл, необхідно вибрати “Перетворити в HTML”. Створений файл *index.html* буде збережений в папці з виконуваним файлом.

# Код

# Реалізація методу DOM API

Основою методу є побудова об’єктної моделі XML-документа. Для цього використовується клас *DOM*\_*XML*\_*Parser*. Відбувається почергове проходження по всіх вузлах нульового рівня. Для кожного з них реалізується зчитування атрибутів цих вузлів. При знаходженні вузлів першого рівня відбувається заглиблення пошуку до атрибутів вузлів першого рівня.

В результаті метод формує список *PeopleList* типу *List<EventItem>* таких випускників, які задовільняють фільтри пошуку *FilterEvent*. Зазначимо, для реалізації усіх трьох методів використовувався інтерфейс *IParser.*

public List<EventItem> Search(FilterEvent filterEvent)

{

var xmlDoc = new XmlDocument();

try { xmlDoc.Load(path); }

catch

{

MessageBox.Show("XML не знайдено!","Помилка");

return null;

}

List<EventItem> PeopleList = new List<EventItem>();

XmlNode root = xmlDoc.DocumentElement;

foreach (XmlNode alumnus in root.ChildNodes) //foreach 0-level-node

{

if (alumnus.NodeType == XmlNodeType.Element)

{

EventItem tmp = new EventItem();

tmp.\_name = (alumnus["name"].InnerText);

tmp.\_faculty = (alumnus["faculty"].InnerText);

tmp.\_cathedra = (alumnus["cathedra"].InnerText);

tmp.\_specialty = (alumnus["specialty"].InnerText);

XmlNode node = alumnus["years"]; //if 1-level-node exists

if (node["entry"] != null) tmp.\_entryYear = node["entry"].InnerText;

if (node["graduation"] != null) tmp.\_graduationYear = node["graduation"].InnerText;

if (Algorythms.Check(tmp, filterEvent)) PeopleList.Add(tmp); //if pupil satisfy search filters -> save pupil

}

}

return PeopleList;

}

# Реалізація методу SAS API

Основою методу є послідовне читання даних з файлу. Для цього використовується клас *SAS\_XML\_Parser.* При зустрічі в файлі кожного вузла нульового рівня викликається метод *ParseItemNode()*, який визначає, які атрибути йдуть після нього (факультет, кафедра і т.д.) і зберігає їх в список *PeopleList*. Аналогічно і для вузлів першого рівня.

public List<EventItem> Search(FilterEvent filterEvent)

{

XmlReader xmlReader;

try { xmlReader = new XmlTextReader(path); }

catch { return null; }

List<EventItem> PeopleList = new List<EventItem>();

try

{

xmlReader.ReadToFollowing("AlumniList");

while (xmlReader.Read())

{

if (xmlReader.Name.Equals("alumnus") && xmlReader.NodeType == XmlNodeType.Element) //find 0-level-node

{

EventItem eI = new EventItem();

eI = ParseItemNode(xmlReader.ReadSubtree()); //parse every alumnus

if (Algorythms.Check(eI, filterEvent)) PeopleList.Add(eI); //if pupil satisfy search filters -> save pupil

}

}

}

catch (Exception e) { throw e; }

finally { xmlReader.Close(); }

return PeopleList;

}

private EventItem ParseItemNode(XmlReader xmlReader)

{

EventItem eI = new EventItem();

while (xmlReader.Read()) //parse every node in the subtree of <alumnus>

{

if (xmlReader.NodeType == XmlNodeType.Element)

{

switch (xmlReader.Name)

{

case "name":

eI.\_name = xmlReader.ReadElementContentAsString();

break;

case "faculty":

eI.\_faculty = xmlReader.ReadElementContentAsString();

break;

case "cathedra":

eI.\_cathedra = xmlReader.ReadElementContentAsString();

break;

case "specialty":

eI.\_specialty = xmlReader.ReadElementContentAsString();

break;

case "years":

ParseYearsNode(xmlReader.ReadSubtree(),ref eI);

break;

}

}

else if (xmlReader.Name.Equals("alumnus") && xmlReader.NodeType == XmlNodeType.EndElement)

{

break;

}

}

return eI;

}

private void ParseYearsNode(XmlReader xmlReader, ref EventItem eI) //determination entry year and graduation year

{

while (xmlReader.Read())

{

if (xmlReader.NodeType == XmlNodeType.Element)

{

switch (xmlReader.Name)

{

case "entry":

eI.\_entryYear = xmlReader.ReadElementContentAsString();

break;

case "graduation":

eI.\_graduationYear = xmlReader.ReadElementContentAsString();

break;

}

}

else if (xmlReader.Name.Equals("place") && xmlReader.NodeType == XmlNodeType.EndElement)

{

break;

}

}

}

# Реалізація методу LINQ to XML

Метод базується на потужностях бібліотеки *System.Xml.Linq.* Для цього використовується клас *LINQ\_TO\_XML\_Parser.* За допомогою LINQ-запиту здійснюється фільтрація елементів таблиці.

public List<EventItem> Search(FilterEvent filterEvent)

{

XDocument doc;

try { doc = XDocument.Load(path); }

catch { MessageBox.Show("XML не знайдено!", "Помилка"); return null; }

List<EventItem> PeopleList = new List<EventItem>();

IEnumerable<EventItem> eventItems = from ev in doc.Descendants("alumnus") //LINQ Query

select new EventItem(

ev.Element("name").Value,

ev.Element("faculty").Value,

ev.Element("cathedra").Value,

ev.Element("specialty").Value,

ev.Element("years").Element("entry").Value,

ev.Element("years").Element("graduation").Value

);

foreach(EventItem eventItem in eventItems)

{

if (Algorythms.Check(eventItem, filterEvent)) PeopleList.Add(eventItem);

}

return PeopleList;

}

# Перетворення XML в HTML

Перетворення здійснювалось шляхом записування тегів і даних у об’єкт типу *StringBuilder* з подальшим створенням файлу *index.html*, в який через потік *Filestream* запишуться всі дані з *StringBuilder.*

private async void MakeHTML()

{

string tab = "\t";

StringBuilder sb = new StringBuilder();

sb.AppendLine("<html>");

sb.AppendLine(tab + "<body>");

sb.AppendLine(tab + tab + "<table>");

// headers.

sb.Append(tab + tab + tab + "<tr>");

foreach (DataColumn dc in table.Columns)

{

sb.AppendFormat("<td>{0}</td>", dc.ColumnName);

}

sb.AppendLine("</tr>");

// data rows

foreach (DataRow dr in table.Rows)

{

sb.Append(tab + tab + tab + "<tr>");

foreach (DataColumn dc in table.Columns)

{

string cellValue = dr[dc] != null ? dr[dc].ToString() : "";

sb.AppendFormat("<td>{0}</td>", cellValue);

}

sb.AppendLine("</tr>");

}

sb.AppendLine(tab + tab + "</table>");

sb.AppendLine(tab + "</body>");

sb.AppendLine("</html>");

string name = "index.html";

FileStream file = File.Open(name, FileMode.Create);

using (StreamWriter outfile = new StreamWriter(file,Encoding.Unicode))

{

await outfile.WriteAsync(sb.ToString());

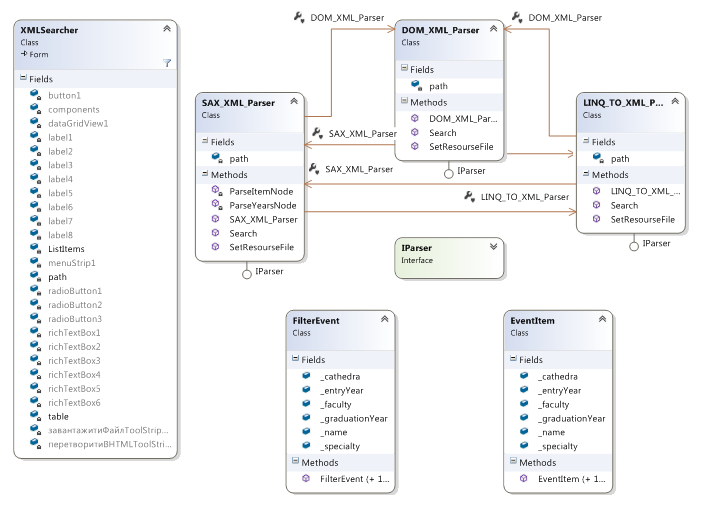
}

MessageBox.Show("Операція здійснена успішно!", "Успіх");

}

}

# Діаграма класів

**

# Список використаних джерел

1. msdn.microsoft.com
2. metanit.com
3. Шилдт Г. - C# 4.0 полное руководство – 2011
4. Шилдт Г. – Полный справочник по С#