УВАГА! ДОКУМЕНТ НАДАНО СТУДЕНТОМ 2 КУРСУ ПОТОКУ ІНФОРМАТИКИ 2014-2015 Н.Р.

Лабораторна робота № 3

Є інформаційна система (вважаємо, що вона вже існує і на даному етапі її розробляти непотрібно!). Інформація з інформаційної системи передається між клієнтами та системою у вигляді XML-документів. Необхідно забезпечити обробку цих документів. Обробка включає в себе дві задачі: аналіз вмісту документу (пошук інформації за ключовими словами та динамічну генерацію запитів) та трансформацію у файл HTML. Вхідні дані для аналізу та трансформації надаються у вигляді файлу-прикладу *.xml. Трансформація документу в HTML-код виконується на основі XSL-документа *.xsl. Аналіз вмісту документа повинен бути виконаний трьома способами – за допомогою SAX API, DOM API та LINQ to XML. Необхідно реалізувати всі три способи!

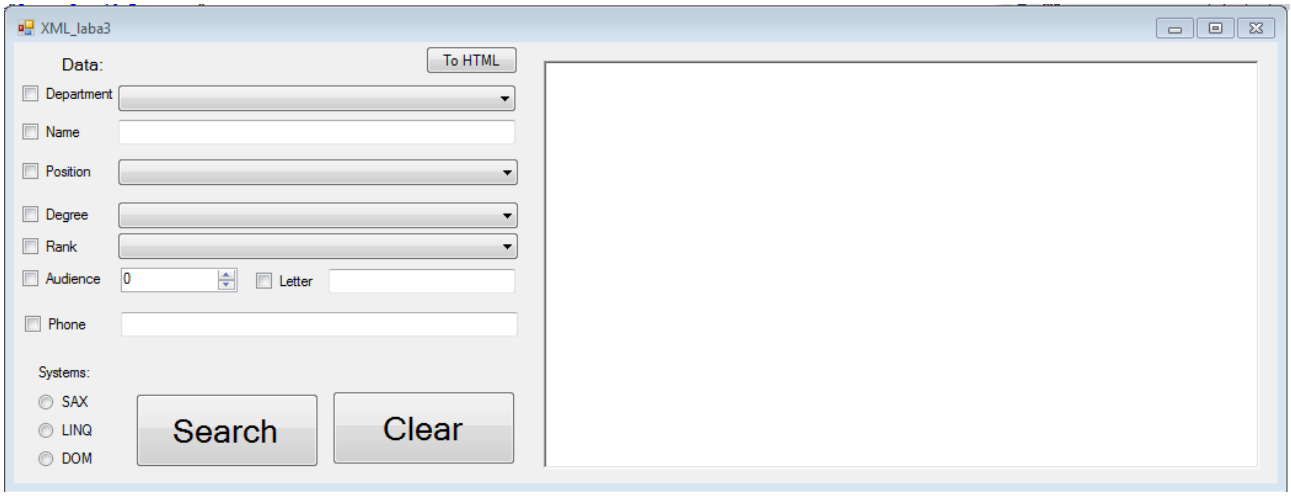
1. Створення XML документу

Прописуємо власноруч текст файлу в редакторі (Наприклад в Блокноті або Notepad++). Зберігаємо в типі *.xml. Потрібно враховувати, що повинен існувати ОДИН головний кореневий об’єкт, який містить всі потрібні нам. В даному випадку база даних про всі кафедри містить кафедри, а кафедри містять викладачів. Атрибути вузлів створюєте на власний смак.

```
<?xml version = "1.0" encoding = "utf-8"?>
<departmensDataBese>
  <department ID ="1" NANE ="Кафедра Обчислювальної математики">
    <employee ID ="1"
              NAME="Ляшко Сергій Іванович"
              POSITION ="завідувач кафедри обчислювальної математики"
              DEGREE ="доктор фізико-математичних наук"
              AUDIENCE ="216"
              PHONE="259-05-32"
              INTERESTS ="інформатика, моделювання та оптимізація систем">
      <rank RANK="професор"></rank>
      <rank RANK="член-кореспондент НАН України"></rank>
    </employee>
    <employee ID ="2"
              NAME="Чикрій Аркадій Олексійович"
              POSITION ="професор"
              DEGREE ="доктор фізико-математичних наук"
              AUDIENCE ="212"
              PHONE="-"
              INTERESTS ="-">
      <rank RANK ="професор"></rank>
      <rank RANK ="член-кореспондент НАН України"></rank>
    </employee>
    <employee ID ="3"
              NAME="Задірака Валерій Костянтинович"
              POSITION ="професор"
              DEGREE ="доктор фізико-математичних наук"
              AUDIENCE ="213"
              PHONE="259-04-36"
              INTERESTS ="-">
      <rank RANK ="професор"></rank>
    </employee>
```

Враховано, що одна людина може мати багато звань. Тому створюються дочірні вузли-звання. Враховуйте такі можливості для своєї роботи.

2. Створення форми



Потрібно враховувати можливість пошуку за кожним критерієм окремо і комбіновано. Тому доцільно використовувати CheckBox.
Оскільки пошук буде вестися лише одною з систем, то доцільно використовувати RadioButton.
В форму не повинно бути вшито даних. Краще на початку виконання програми прочитати всі можливі варіанти параметрів і задати в ComboBox. Так ви будете знати, з чого вибирати, без ризику зробити помилку.

```
private void MainWindow_Load(object sender, EventArgs e)
{
    AddDepartments();
    AddPositions();
    AddDegree();
    AddRank();
}
1 reference
private void AddDepartments()
{
    string[] s = Finder.GetAllDepartments();
    if(s!=null)
    {
        comboBox1.Items.Clear();
        foreach(string st in s )
        {
            comboBox1.Items.Add(st);
        }
    }
}
```

Потрібно прибрати можливість вводити самому дані, а дозволити лише вибирати.
ComboBox -> Properties->DropDownStyle->DropDownList

Форма не повинна працювати з файлами чи робити операції над даними. Для цього слід виділити окремий клас (Finder в даному випадку) і також клас, який буде слугувати контейнером для обміну інформацією між формою та моделлю(Emploee (трошки не дописав - буває) в даному випадку). За таких умов формі все рівно, звідки беруться дані, її завдання:

- Передати параметри пошуку
- Отримати дані та відобразити їх

Тоді запит форми буде виглядати приблизно так:

```
private void searchButton1_Click(object sender, EventArgs e)
{
    resultRichTextBox1.Clear();
    if (domRadioButton1.Checked)
    {
        List<Emploee> r = Finder.SearchByDOM(ReadParameters());
        Emploee[] t = r.ToArray();
        foreach (Emploee empl in t)
        {
            ShowEmploee(empl);
        }
    }
    if(linqRadioButton1.Checked)
    {
        List<Emploee> r = Finder.SearchByLINQ(ReadParameters());
        Emploee[] t = r.ToArray();
        foreach (Emploee empl in t)
        {
            ShowEmploee(empl);
        }
    }
    if(saxRadioButton1.Checked)
    {
        List<Emploee> r = Finder.SearchBySAX(ReadParameters());
        Emploee[] t = r.ToArray();
        foreach (Emploee empl in t)
        {
            ShowEmploee(empl);
        }
    }
}
```

«Контейнер» виглядає так:

```
class Emploee
{
    15 references
    public string Department { get; set; }
    15 references
    public string Name { get; set; }
    14 references
    public string Position { get; set; }
    14 references
    public string Degree { get; set; }
    public List<string> rank = new List<string>();
    string audience;
    string letter = String.Empty;
    string phone;
    14 references
    public string Interests { set; get; }
    5 references
    public Emploee()
    {
        Department = String.Empty;
        Name = String.Empty;
        Position = String.Empty;
        Degree = String.Empty;
        Interests = String.Empty;
    }
    5 references
    public void SetAudience(string aud,string lett)...
    7 references
    public string GetNumberOfAudience()...
    6 references
    public string GetLetterOfAudience()...
    4 references
    public void SetPhone(string phone)...
    10 references
    public string GetPhone()...
    7 references
    static public string GetOnlyNumbersFromPhone(string phone)...
    1 reference
    public bool Eq(Emploee e)...

}
```

3. Пошук за допомогою DOM

В пошук за допомогою цієї системи я поклав принцип: шукаю за кожним параметром окремо, а потім беру перетин результатів. Такий підхід допустимий лише при не дуже великих об’ємах інформації. Так виглядає головна функція пошуку:

```
static public List<Emploee> SearchByDOM(Emploee empl)
{
    List<List<Emploee>> allResults = new List<List<Emploee>>();
     try
    {
        XmlDocument xmlDoc = new XmlDocument();
        xmlDoc.Load(@"C:\Users\nazar\Documents\Visual Studio 2013\Projects\Laba3\Base.xml");
        allResults.Add(SearchByDomDepartment(empl.Department, xmlDoc));
        allResults.Add(SearchByDomName(empl.Name, xmlDoc));
        allResults.Add(SearchByDomPosition(empl.Position, xmlDoc));
        allResults.Add(SearchByDomDegree(empl.Degree, xmlDoc));
        allResults.Add(SearchByDomAudience(empl.GetNumberOfAudience(), empl.GetLetterOfAudience(), xmlDoc));
        allResults.Add(SearchByDomPhone(empl.GetPhone(), xmlDoc));
        allResults.Add(SearchByDomInterests(empl.Interests, xmlDoc));
        if (empl.rank.Count > 0) { allResults.Add(SearchByDomRank((empl.rank.ToArray())[0], xmlDoc)); }
        else
        {
            allResults.Add(SearchByDomRank(String.Empty, xmlDoc));
        }
    }
    catch { }
    return CrossingOfResults(allResults);
}
```

Всі фільтри абсолютно ідентичні. Ось приклад одного з них:

```
static public List<Emploee> SearchByDomName(string param , XmlDocument xmlDoc)
{
    List<Emploee> r = new List<Emploee>();
    if (param != null)
    {
        if (param != String.Empty)
        {
            XmlNodeList elem = xmlDoc.SelectNodes("//employee[@NAME=\"" + param + "\"]");
            try
            {
                foreach (XmlNode e in elem)
                {
                    r.Add(GetDataAboutEmploee(e));
                }
            }
            catch { }
            return r;
        }
    }
    return GiveAllEmploeeDOM(xmlDoc.DocumentElement);
}
```

Тепер потрібно перетнути результати:

```
static private List<Emploee> CrossingOfResults(List<List<Emploee>> list)
{
    List<Emploee> result = new List<Emploee>();
    try
    {
        if (list != null)
        {
            Emploee[] em = list[0].ToArray();
            if (em != null)
            {
                foreach (Emploee ele in em)
                {
                    bool isIN = true;
                    foreach (List<Emploee> t in list)
                    {
                        if (t.Count <= 0) { return new List<Emploee>(); }
                        foreach (Emploee e in t)
                        {
                            isIN = false;
                            if (ele.Eq(e)) { isIN = true; break; }
                        }
                        if (isIN == false) { break; }
                    }
                    if (isIN == true) { result.Add(ele); }
                }
            }
        }
    }
    catch { }
    return result;
}
```

Результат передається формі.

4. Пошук за допомогою LINQ

В даному випадку не важко написати запит, який фільтрує по всім параметрам одночасно.

```
static public List<Emploee> SearchByLINQ(Emploee empl)
{
    List<Emploee> results = new List<Emploee>();
    XDocument docXML = XDocument.Load(@"C:\Users\nazar\Documents\Visual Studio 2013\Projects\Laba3\Base.xml");
    var result = (from obj in docXML.Descendants("employee")
                  where (

                  (empl.Name == String.Empty || empl.Name == obj.Attribute("NAME").Value) &&
                  (empl.Position == String.Empty || empl.Position == obj.Attribute("POSITION").Value) &&
                  (empl.Degree == String.Empty || empl.Degree == obj.Attribute("DEGREE").Value) &&
                  ((empl.GetLetterOfAudience() == CutNumbers(obj.Attribute("AUDIENCE").Value) &&
                  empl.GetNumberOfAudience() == Emploee.GetOnlyNumbersFromPhone(obj.Attribute("AUDIENCE").Value)) ||
                  empl.GetNumberOfAudience() == String.Empty) &&
                  (empl.GetPhone() == String.Empty || empl.GetPhone() == Emploee.GetOnlyNumbersFromPhone( obj.Attribute("PHONE").Value)) &&
                  (empl.Interests == String.Empty || empl.Interests == obj.Attribute("INTERESTS").Value) &&
                  (empl.Department == String.Empty || empl.Department == obj.Parent.Attribute("NANE").Value) &&
                  (empl.rank.Count <= 0 || obj.Descendants().Any(elem => (elem.Attribute("RANK").Value == empl.rank[0])))

                  )

                  select obj
                              ).ToList();
```

Але потрібно пам’ятати, що пошук не відбувається одразу. Тому існування контейнеру грає нам на руку. Переписуємо результати в прийнятному для форми вигляді:

```
                  select obj
                              ).ToList();

    foreach(var r in result)
    {
        Emploee e = new Emploee();
        e.Name = r.Attribute("NAME").Value;
        e.Position = r.Attribute("POSITION").Value;
        e.Degree = r.Attribute("DEGREE").Value;
        e.Department = r.Parent.Attribute("NANE").Value;
        e.SetAudience(Emploee.GetOnlyNumbersFromPhone(r.Attribute("AUDIENCE").Value), CutNumbers(r.Attribute("AUDIENCE").Value));
        e.SetPhone(r.Attribute("PHONE").Value);
        e.Interests = r.Attribute("INTERESTS").Value;
        var ranks = r.Descendants();
        foreach(var rank in ranks)
        {
            e.rank.Add(rank.Attribute("RANK").Value);
        }
        results.Add(e);
    }

    return results;
}
```

Пошук реалізовано єдиним методом.

5. Пошук за допомогою SAX

В даному випадку я вирішив дістати з xml-документу всіх робочих, а потім відфільтрувати за параметром.
Зчитування має наступний вигляд:

```
static public List<Emploee> SearchBySAX(Emploee empl)
{
    List<Emploee> results = new List<Emploee>();
    string dep = String.Empty;
    var xmlReader = new XmlTextReader(@"C:\Users\nazar\Documents\Visual Studio 2013\Projects\Laba3\Base.xml");
    Emploee betw = null;
    while(xmlReader.Read())
    {
        if (xmlReader.Name == "department") { while (xmlReader.MoveToNextAttribute())
        { if (xmlReader.Name == "NANE") { dep = xmlReader.Value;

        } }
        }
        if(xmlReader.Name == "employee")
        {
            if(betw == null)
            {
                betw = new Emploee();
                betw.Department = dep;
            }
            else
            {
                if(betw.Name!=String.Empty)
                results.Add(betw);
                betw = new Emploee();
                betw.Department = dep;
            }
    if(xmlReader.HasAttributes)
    while(xmlReader.MoveToNextAttribute())
    {
        if (xmlReader.Name == "NAME") {if(xmlReader.HasValue) betw.Name = xmlReader.Value; }
        if (xmlReader.Name == "DEGREE") { if (xmlReader.HasValue) betw.Degree = xmlReader.Value; }
        if (xmlReader.Name == "POSITION") { if (xmlReader.HasValue) betw.Position = xmlReader.Value; }
        if (xmlReader.Name == "INTERESTS") { if (xmlReader.HasValue) betw.Interests = xmlReader.Value; }
        if (xmlReader.Name == "PHONE") { if (xmlReader.HasValue)betw.SetPhone(xmlReader.Value); }
        if (xmlReader.Name == "AUDIENCE") { if (xmlReader.HasValue) betw.SetAudience(Emploee.GetOnlyNumbersFromPhone(xmlReader.Value), CutNumbers(xmlReader.Value)); }
    }
}


    if (xmlReader.Name == "rank")
    {
        while (xmlReader.MoveToNextAttribute())
        {
            if (xmlReader.HasValue)
                betw.rank.Add(xmlReader.Value);
        }
    }

}
```

Тепер потрібно відфільтрувати і повернути результати.
Краще створити окрему функцію-фільтр:

```
static List<Emploee> FiltrByEmployee(List<Emploee> allEmpl, Emploee param)
{
    List<Emploee> result = new List<Emploee>();
    if(allEmpl!=null)
    {
        foreach(Emploee e in allEmpl)
        {
            try
            {
                if ((e.Name == param.Name || param.Name == String.Empty) &&
                    (e.Interests == param.Interests || param.Interests == String.Empty) &&
                    (e.Position == param.Position || param.Position == String.Empty) &&
                    (e.Department == param.Department || param.Department == String.Empty) &&
                    (e.Degree == param.Degree || param.Degree == String.Empty) &&
                    (e.GetPhone() == param.GetPhone() || param.GetPhone() == String.Empty) &&
                    ((e.GetLetterOfAudience() == param.GetLetterOfAudience() && e.GetNumberOfAudience() == param.GetNumberOfAudience()) || (param.GetNumberOfAudience() == String.Empty))

                    )
                {
                    if (e.rank.Count <= 0 || param.rank.Count <=0)
                    {
                        result.Add(e);
                    }
                    else
                    {
                        if(e.rank.Contains(param.rank[0]))
                            result.Add(e);
                    }
                }
            }
            catch { }
        }
    }
    return result;
}
```

Тоді залишається записати в попередню функцію:

```
return FiltrByEmployee(results,empl);
```

Форма отримує результат.

6. Трансформація

Найголовніше на цьому етапі правильно написати файл трансформації, враховуючи структуру свого xml.

Приклад xsl:

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method= "html"/>
  <xsl:template match="department">
    <h1><xsl:value-of select="@NANE"/></h1>
    <TABLE BORDER ="5">
      <TR>
    <xsl:for-each select="employee">
      <TR>
        <TD>
      <TABLE BORDER ="1" WIDTH ="1200">
        <TR>
        <th style="width:10%;">
          <p align ="left">ПІП</p>
        </th>
        <th style="width:90%;">
          <p align ="left">
            <xsl:value-of select="@NAME"/>
          </p>
        </th>
      </TR>
        <TR>
        <th style="width:10%;">
          <p align ="left">Посада</p>
        </th>
        <th style="width:90%;">
          <p align ="left">
            <xsl:value-of select="@POSITION"/>
          </p>
        </th>
      </TR>
        <TR>
          <th style="width:10%;">
            <p align ="left">Освіта</p>
```

Логіка трансформування. Для кожного працівника створюється таблиця з даними, вона міститься в загальній таблиці кафедри. Для кожної кафедри створюється така таблиця.

Звернути увагу на:

- співвідношення між розмірами комірок

```
<th style="width:90%;">
```

(без співвідношення розміри комірок залежать від кількості даних, а для різних працівників вона, як правило, різна – вийде «крива» таблиця)

- вирівнювання

```
<p align ="left">
```

(без нього дані будуть в центрі комірки – теж не гарно).

Результат:

Кафедра Обчислювальної математики

ПІП	Ляшко Сергій Іванович
Посада	завідувач кафедри обчислювальної математики
Освіта	доктор фізико-математичних наук
Звання	професор; член-кореспондент НАН України;
Аудиторія	216
Телефон	259-05-32
Інтереси	інформатика, моделювання та оптимізація систем
ПІП	Чикрій Аркадій Олексійович
Посада	професор
Освіта	доктор фізико-математичних наук
Звання	професор; член-кореспондент НАН України;
Аудиторія	212
Телефон	-
Інтереси	-
ПІП	Задірака Валерій Костянтинович
Посада	професор
Освіта	доктор фізико-математичних наук
Звання	професор;
Аудиторія	213
Телефон	259-04-36
Інтереси	-

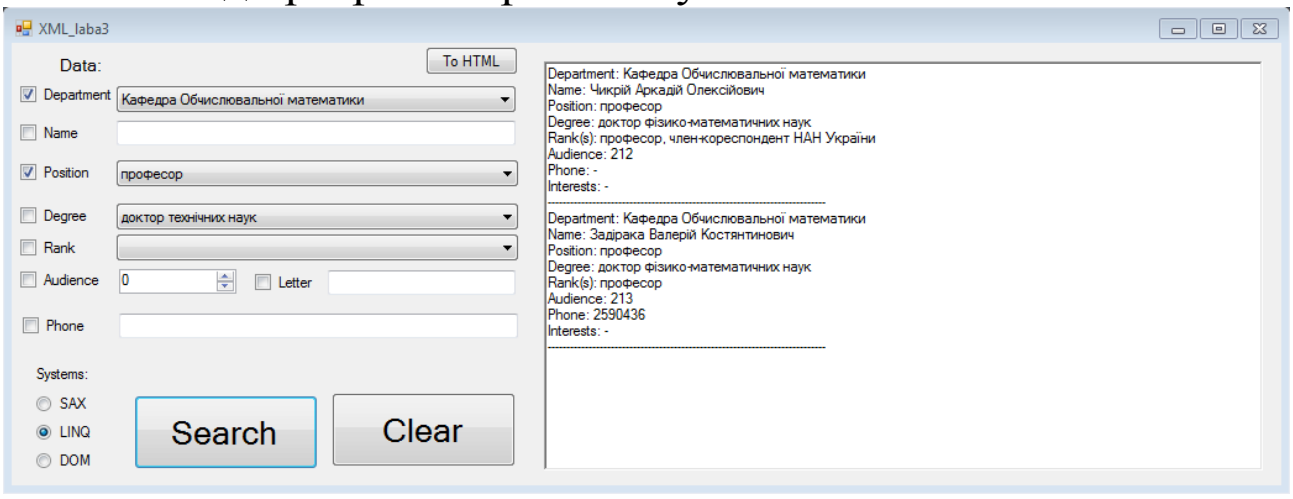
Кафедра Моделювання складних систем

ПІП	Гаращенко Федір Георгійович
Посада	завідувач кафедри моделювання складних систем
Освіта	доктор технічних наук
Звання	професор; академік АН ВШ України;
Аудиторія	410
Телефон	-
	якісний аналіз та оцінка програмних траєкторій в системах керування; дослідження задач практичної стійкості динамічних систем та розробка

Функція конвертації оформлюється так:

```
static public void Transform()
{
    XslCompiledTransform transf = new XslCompiledTransform();
    transf.Load(@"C:\Users\nazar\Documents\Visual Studio 2013\Projects\Laba3\transformation.xsl");
    transf.Transform(@"C:\Users\nazar\Documents\Visual Studio 2013\Projects\Laba3\Base.xml", @"C:\Users\nazar\Documents\Visual Studio 2013\Projects\Laba3\newBase.html");
}
```

7. Вигляд програми в робочому стані:



Увага: Як бачимо, «Освіта» не врахована, оскільки ми не поставили «галочку»