2016

Милка Ферезлийска №61710 гр.2

XML технологии за семантичен уеб

1/1/2016



Бланка за кандидатстване

за избираеми дисциплини

1. Условие

Изискванията към системата са да предоставя възможност за разглеждане и записване на избираеми дисциплини от ФМИ през няколко форми. Попълнените данни да се съхраняват и четат от .xml документ. Съхранените данни трябва да могат да се валидират с XSD. Грешките при валидацията трябва да се съхраняват в отделен файл, чието съдържание да се изчиства преди всяко стартиране на програмата.

1. Въведение

Разработена е система, отговаряща на изискванията от точка 1 в документа като десктоп приложение, написано на C#.

Системата предоставя възможност за разглеждане на списък от избираеми дисциплини във ФМИ. На вниманието на потребителите е предложен списък с такива. Освен преглед на дисциплините, потребителят може и да се запише за произволен брой от тях.

Поради естеството на системата, се изисква попълване на лични данни на студента като имена, телефон, адрес на местоживеене, факултетен номер, курс и специалност. Попълнените данни се съхраняват в .xml документ. След записването на данните в документа, той се валидира от XSD, с която е свързан. Ако се открият грешки при валидацията на съдържанието му, те се записват в нов файл на име error\_log.txt. Системата проверява за съществуването на файл с такова име и ако съществува, изтрива съдържанието му, а ако не съществува – го създава, като и в двата случая променя съдържанието му с грешките от текущата валидация на създадения XML документ.

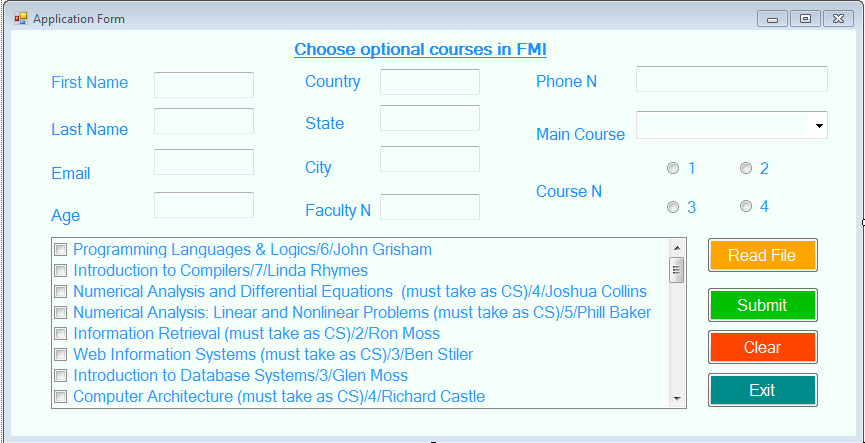
1. Архитектура на приложението

Системата се състои от две форми за взаимодействие с потребителите.

Главната форма се нарича **appForm** и съдържа в себе си всички полета за въвеждане на потребителските данни. За да запише курс, един потребител трябва да въведе своите имена, телефон, адрес, факултетен номер, курс, специалност и адрес. Задължителен е изборът на курс, ако иска да създаде нов запис във файла. Когато потребителят се опита да запази данните от формата без да е избрал нито един курс, се извежда съобщение за грешка и формата не изчиства данните си. Функционалността е имплементирано по този начин, за да бъде приложението лесно за използване от потребителите.

Освен текстовите полета, радио и чек бокс бутоните, на тази форма са разположени и няколко бутона за управлението й.

Формата се запазва при натискане на бутона **Submit**. Той извършва проверка за съществуването на документ с данни за записани избираеми дисциплини. Ако файлът съществува, го допълва, а ако не – го създава и написва съдържание. След успешно записване на данните от формата, тя се изчиства.

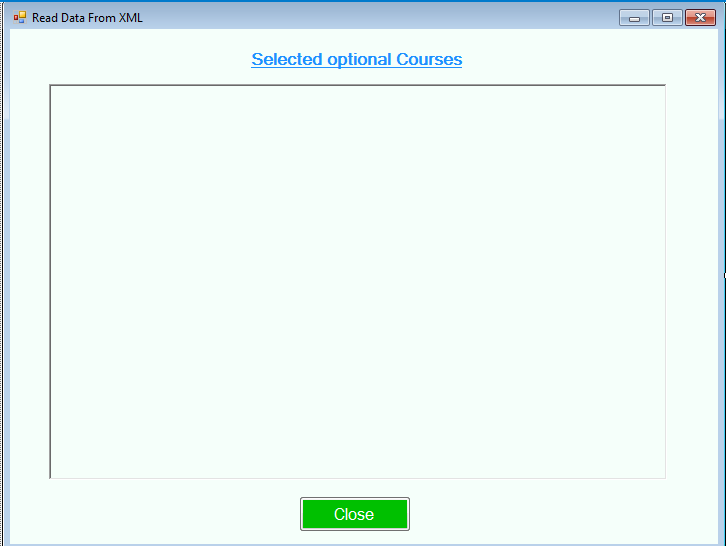


Друг важен бутон, разположен на главната форма, е Clear бутонът. Неговата функция е да изчисти всички полета от формата. Автоматично се натиска и при успешно запазване на данните в документа след **Submit**.

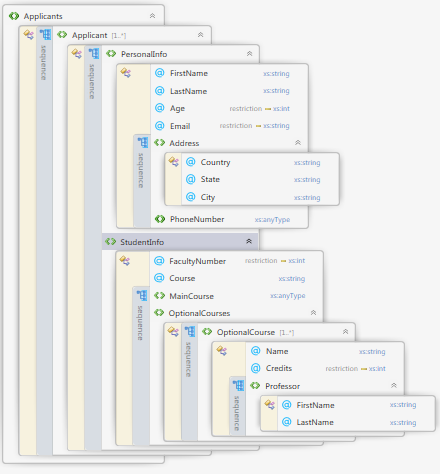
**Exit** бутонът затваря приложението.

Функционалността за четене на данни от съществуващ XML документ, генериран предварително с приложението, е налична за бутона **Read File**. При натискането му, се зареждат данните от документа в паметта и се отваря нова форма, в която се визуализират.

Втората форма носи името **readForm** и нейната роля е само и единствено да покаже данните от файла. В нея е разположено текстово поле за четене на данните и бутон за затварянето й, който препраща потребителя към главната форма.



Генерираният XML документ има следната структура:



1. Използвани технологии

В системата са използвани два типа на представяне на XML документа – **XmlDocument** и **XDocument**. И двата класа предоставят възможност за зареждане на XML документ в паметта и работа с него. С **XmlDocument** се създава и чете документа, а с **XDocument** се валидира съществуващия файл.

doc = new XmlDocument();

XmlElement applicants = doc.CreateElement("Applicants");

XmlElement applicant = doc.CreateElement("Applicant");

applicants.AppendChild(applicant);

doc.AppendChild(applicants);

doc.Save(@".\created\_doc.xml");

Елементите се създават с помощта на **XmlElement**, а четенето им от документа, се осъществява посредством **XmlNode**:

doc.Load(xml);

XmlNode applicants = doc.FirstChild;

foreach (XmlNode applicant in applicants.ChildNodes)

{

XmlNode personalInfo = applicant.FirstChild;

}

1. Инсталация и настройки

Приложението се използва слез сваляне и разархивиране на архива с файловете при отваряне на файла **ApplicationForm.exe**. Не са необходими допълнителни настройки.

1. Примерни данни

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

<Applicants>

<Applicant>

<PersonalInfo FirstName="Ivan" LastName="Georgiev" Age="17" Email="lqlq@ldl.com">

<Address Country="Test Country" State="Test Sate" City="Test City"/>

<PhoneNumber>5558989</PhoneNumber>

</PersonalInfo>

<StudentInfo FacultyNumber="59342" Course="3">

<MainCourse>Software Engineering</MainCourse>

<OptionalCourses>

<OptionalCourse Name="OOP" Credits="5">

<Proffessor FirstName="Ivan" LastName="Gecov"></Proffessor>

</OptionalCourse>

<OptionalCourse Name="Java Script" Credits="6">

<Proffessor FirstName="Minko" LastName="Minchev"></Proffessor>

</OptionalCourse>

</OptionalCourses>

</StudentInfo>

</Applicant>

</Applicants>

1. Кратко ръководство на потребителя

Потребителят стартира приложението от файла **ApplicationForm.exe.** Ако иска да запише нов курс, попълва формата и натиска бутона **Submit**. Ако иска само да разгледа документа със записаните курсове, натиска **Read File**. Ако иска просто да изтрие попълнената форма или пък да я затвори - съответно **Clear** и **Close**.

1. Описание на програмния код

Програмата зарежда в паметта нов обект от тип **XmlDocument**, който моделира съществуващият или файла, който ще се създаде и ще съхранява данните от приложението. При натискане на различни части от потребителския интерфейс, се случват поредица от събития, водещи до очаквания резултат. Подробности за описанието на бутоните има по-горе в документа.

Събмитването на кода се приема от тази функция, която проверява валидацията и съществуването на XML файл:

private void submit\_Click(object sender, EventArgs e)

{

createFile();

if (validate())

{

this.clear.PerformClick();

this.readFile.Enabled = true;

}

}

Създаването на файл се извършва от следната функция, която създава в паметта структурата и данните, попълнени във формата и ги запазва в променливата **doc**. Когато всички елементи със съответните атрибути се закачат за него, той се запазва в XML документа. Като проверява дали файлът съществува. Ако съществува, дописва отдолу данните си, а ако не съществува, го създава и напълва с информация:

private void createFile()

{

XmlElement applicants = doc.CreateElement("Applicants");

XmlElement applicant = doc.CreateElement("Applicant");

XmlElement personalInfo = doc.CreateElement("PersonalInfo");

personalInfo.SetAttribute("FirstName", this.firstName.Text);

personalInfo.SetAttribute("LastName", this.lastName.Text);

personalInfo.SetAttribute("Age", this.age.Text);

personalInfo.SetAttribute("Email", this.email.Text);

XmlElement address = doc.CreateElement("Address");

address.SetAttribute("Country", this.country.Text);

address.SetAttribute("State", this.state.Text);

address.SetAttribute("City", this.city.Text);

XmlElement phoneNumber = doc.CreateElement("PhoneNumber");

phoneNumber.InnerText = this.phoneNumber.Text;

personalInfo.AppendChild(address);

personalInfo.AppendChild(phoneNumber);

XmlElement studentInfo = doc.CreateElement("StudentInfo");

studentInfo.SetAttribute("FacultyNumber", this.facNumber.Text);

studentInfo.SetAttribute("Course", this.getCourse());

XmlElement mainCourse = doc.CreateElement("MainCourse");

mainCourse.InnerText = this.mainCourse.Text;

XmlElement optionalCourses = doc.CreateElement("OptionalCourses");

foreach(String optCourse in this.optionalCourses.CheckedItems) {

XmlElement optionalCourse = doc.CreateElement("OptionalCourse");

String[] text = optCourse.Split('/');

String credits = text[1];

String[] names = text[2].Split(' ');

optionalCourse.SetAttribute("Name", text[0]);

optionalCourse.SetAttribute("Credits", credits);

XmlElement professor = doc.CreateElement("Professor");

professor.SetAttribute("FirstName", names[0]);

professor.SetAttribute("LastName", names[1]);

optionalCourse.AppendChild(professor);

optionalCourses.AppendChild(optionalCourse);

}

if (this.optionalCourses.CheckedItems.Count > 0)

{

this.clearCheckboxes();

}

else

{

ErrorProvider e = new ErrorProvider();

e.SetError(this.optionalCourses, "You must select at least one course!");

return;

}

studentInfo.AppendChild(mainCourse);

studentInfo.AppendChild(optionalCourses);

applicant.AppendChild(personalInfo);

applicant.AppendChild(studentInfo);

applicants.AppendChild(applicant);

if (File.Exists(xml))

{

doc.Load(xml);

XmlNode root = doc.FirstChild;

root.AppendChild(applicant);

}

else

{

doc.AppendChild(applicants);

}

doc.Save(xml);

}

Функцията за валидация на генерирания XML документ се нарича **validate** и се извиква при натискане на **Submit**:

private bool validate()

{

XmlSchemaSet schemas = new XmlSchemaSet();

schemas.Add("", xsd);

if (!File.Exists(xml))

return false;

XDocument document = XDocument.Load(xml);

bool hasErrors = false;

document.Validate(schemas, (output, error) =>

{

Console.WriteLine("{0}", error.Message);

hasErrors = true;

if (hasErrors)

{

using (StreamWriter sw = File.AppendText(error\_log))

{

sw.WriteLine(String.Format(

"{0}: {1}",

error.Severity,

error.Message

));

}

}

});

Console.WriteLine("doc1 {0}", hasErrors ? "did not validate" : "validated");

return hasErrors;

}

1. Приноси на студента, ограничения и възможности за бъдещо разширение

Съществуващата програма може да се рефактурира и да се използва по-новият клас за зареждане на XML документи в паметта **XDocument**. Работата с него е по-улеснена и бърза и това ще намали броя редове и излишен, повтарящ се код в системата. Също така може да се промени начинът на визуализация на прочетените данни като се използва **XSLT** и табличен формат.

1. Какво научих

Създаването на тази система ми помогна да организирам и систематизирам знанията си, свързани с XML и C#, за да направя работещо приложение. Успях да приложа на практика и знанията, които придобих по време на упражненията, в цялостен проект.

1. Използвани източници

* Whitespace Handling in XSLT Transformation, © 2016 Microsoft, 23.01.2016 20:30, [MSDN - Whitespace Handling](https://msdn.microsoft.com/en-us/library/aa468566.aspx)
* XmlDocument Class, © 2016 Microsoft, 10.01.2016 19:54, [MSDN - XmlDocument Class](https://msdn.microsoft.com/en-us/library/system.xml.xmldocument%28v=vs.110%29.aspx)
* XmlNode Class, © 2016 Microsoft, 10.01.2016 20:49, [MSDN - XMLNode Class](https://msdn.microsoft.com/en-us/library/system.xml.xmlnode%28v=vs.110%29.aspx)
* XDocument Class, © 2016 Microsoft, 10.01.2016 21:27, [MSDN - XDocument Class](https://msdn.microsoft.com/en-us/library/system.xml.linq.xdocument%28v=vs.110%29.aspx)
* XDocument.Load Method (String), © 2016 Microsoft, 10.01.2016 23:52, [MSDN - XDocument.Load Method](https://msdn.microsoft.com/en-us/library/bb343181%28v=vs.110%29.aspx)
* Understanding XML Schema, © 2016 Microsoft, 12.01.2016 19:16, [MSDN - XML Schema](https://msdn.microsoft.com/en-us/library/aa468557.aspx)
* XmlSchema Class, © 2016 Microsoft, 12.01.2016 19:27, [MSDN - XmlSchema Class](https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschema%28v=vs.110%29.aspx)