

Програмиране за разпределени среди

Курсова работа

**„Функционалност, специфична за JSON формата,
която няма директен аналог в XML“**

Съставил: Ангел Любомиров Стойнов

Факултетен номер: 121222150

Група: 40

Съдържание

Детайлно задание	3
JSON	4
XML	5
Специфична функционалност разглеждана в курсовата работа.....	6
Разработка	7
General Overview	7
Файлова структура	7
Описание по разработката	9

Детайлно задание

Функционалност, специфична за JSON формата, която няма директен аналог в XML. Функционалността е по ваш избор.

- a) Намерете функционалност свързана с JSON , която няма директен аналог в XML
 - а. не че не може да се направи с XML, но не може под същата форма
- b) Намерете готов JSON файл или създайте подходящ JSON файл с достатъчно данни, върху който може да се демонстрира функционалността (от т.а).
- c) Създайте програма (на среда .Net), която да изпълнява функционалността (от т.а) върху демонстрационния файл (от т.б)
- d) Създайте еквивалентен XML файл на демонстрационния файл (от т.2)
- e) Създайте програма (на среда .Net), която да изпълнява функционалността (от т.а) върху еквивалентния файл (от т.е)
- f) Документирайте разликата и ефекта от разликата. Документирайте всичко необходимо за да изпълните предните точки.

Всяка курсова работа трябва да съдържа:

Проект на .Net, който включва примери, демонстрации избраната техниката/технологията, на C#. XML/JSON файл, с подходящо съдържание за демонстрираната технология (ако заданието изисква)

Документация (3-4 стр):

Кратко представяне на техниката (вие какво сте разбрали, не какво пише във Wiki, или на вас коя информация ви помогна да разберете, не всичко което LLM/GPT може да каже по темата)

Списък източници, от които сте се запознали (прилагате отговорите от LLM/GPT, ако сте ползвали тях, но е хубаво също и да го накарате той да ви даде източниците които е ползвал) (1-10 стр)

Описание на двата реализирани примера

Защита защо примера е смислен (реален) според вас

JSON

JSON (JavaScript Object Notation) е опростен формат за обмяна на данни. Той е базиран на едно подмножество на езика за програмиране JavaScript, Standard ECMA-262 3rd Edition - от декември 1999 г. JSON има текстов формат, напълно независим от реализацията на езика, но използва конвенции, които са познати на програмистите на C-подобни езици, включително C, C++, C#, Java, JavaScript, Perl, Python, и много други. Тези свойства правят JSON идеален формат за обмяна на данни.

JSON се състои от две структури:

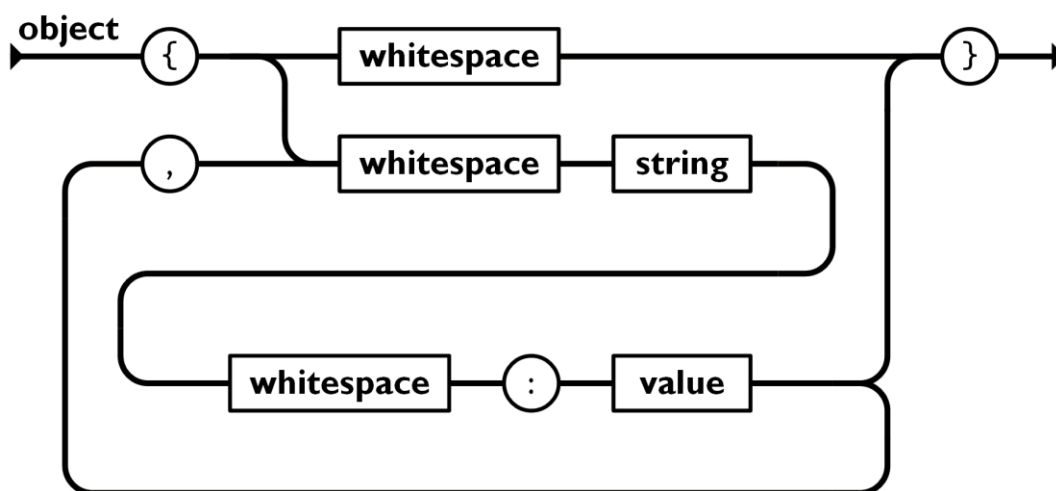
- Колекция от двойки име/стойност (key/pair).
- Подреден списък от стойности.

Обектът започва с **U+007B** ({) и се приключва с **U+007D** (}). Ключовете са в **U+0022** (“) и завършват с **U+003A** (:). Отделните записи за разделени с **U+002C** (,) [1]. Показано на фигура 1 и 2.

Една стойност може да бъде string (низ) в двойни кавички, число, boolean (true/false), null, обект или масив [2]. Тези структури могат да бъдат вложени. [3] [4]

```
1 {  
2   "name": "Angel",  
3   "age": 22,  
4   "universityStudent": true  
5 }
```

На фигура 1 е показан примерен JSON файл.



На фигура 2 е показан структурата на JSON файл. [2]

XML

Extensible Markup Language (XML) представлява универсален формат за описание и съхранение на данни, който улеснява обмена на информация между различни компютърни системи — като уебсайтове, бази данни и външни приложения.

Благодарение на предварително дефинираните синтактични правила, XML осигурява лесно и надеждно предаване на данни през всякакъв тип мрежа, тъй като получателят може точно да интерпретира съдържанието според тези правила. [5]

XML файловете могат да съдържат метаданни, които описват структурата, значението или контекста на самите данни. XML често се използва като основа за различни комуникационни и авторизиращи протоколи, като например SAML (Security Assertion Markup Language), SOAP и други. [6] [7]

Тези протоколи използват XML за формализиране и сигурен обмен на данни между приложения и системи в уеб среда. XML е доста по-вербозен отколкото JSON. Синтаксисът на XML е доста подобен на HTML (фиг. 3). Състои се от отварящи и затварящи тагове, които не са предварително дефинирани. Те се създават от програмиста. Таговете могат да съдържат вложени (дъщерни) елементи, което позволява изграждане на йерархична структура на данните. [5]

Всеки XML документ започва с декларация, която указва версията и кодирането, например [4]:

```
1      <?xml version="1.0" encoding="UTF-8"?>
2      <person>
3          <name>Angel</name>
4          <age>22</age>
5          <universityStudent>true</universityStudent>
6      </person>
```

На фигура 3 е показан синтаксиса на XML. Всички тагове са създадени от програмиста и не са стандартно дефинирани от самия markup language.

Специфична функционалност разглеждана в курсовата работа

В основата си XML не поддържа примитивни типове, всеки елемент се разглежда и третира като string (низ) [5]. За да бъде уточнен примитивен тип (number, boolean и т.н.) е необходимо програмно да се добави атрибута type, показано на фигура 4 [8]. Това допълнително утежнява файла, особено, ако файлът е дълъг и комплексен. По-практично решение на този проблем е използването на XSD – XML Schema Definition, който описва структурата и типовете данни (фиг. 5) [9] [10]. По този начин XML може да постигне ниво на типизация, подобно на JSON [4] [11], но не по подразбиране, а чрез допълнителна схема или логика на приложението.

```
1      <?xml version="1.0" encoding="UTF-8"?>
2      <root>
3          <name type="string">Angel</name>
4          <age type="number">22</age>
5          <universityStudent type="boolean">true</universityStudent>
6      </root>
```

На фигура 4 е показан начин за дефиниране на тип в XML.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

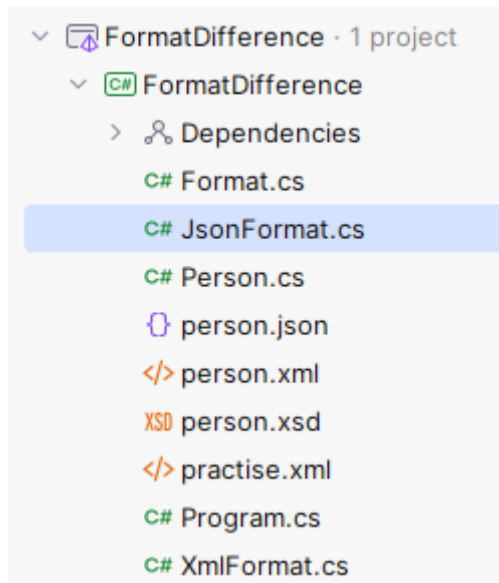
На фигура 5 е показан XSD начин за дефиниране на тип.

Разработка

General Overview

За демонстрация на посочената функционалността е разработено конзолно .Net приложение. Целта му е да покаже различията на JSON и XML формата. Съдържа примери и как може да се имплементира подобна функционалност на XML. Всички **.xml** и **.json** файлове са създадени предварително и могат да бъдат намерени в solution.

Файлова структура



На фигура 6 е показана структурата на приложение. Основно ще се наблегне върху JsonFormat.cs, XmlFormat.cs, Person.cs и Program.cs

JsonFormat.cs

```
7 ^o public class JsonFormat : Format
8 {
9     private const string JsonPath = "person.json";
10
11 ^o public override void Display()
12 {
13     var path = Path.Combine(AppContext.BaseDirectory, JsonPath);
14     if (!File.Exists(path))
15     {
16         Console.WriteLine("File not found.");
17         return;
18     }
19
20     var json:string = File.ReadAllText(path);
21     var person = JsonSerializer.Deserialize<Person>(json!);
22
23     Console.WriteLine("Name: " + person.name + " Type: " + person.name.GetType());
24     Console.WriteLine("Age: " + person.age + " Type: " + person.age.GetType());
25     Console.WriteLine("Status: " + person.universityStudent + " Type: " + person.universityStudent.GetType());
26 }
27 }
```


Описание по разработката

Цитирани източници

- [1] „Unicode“, [Онлайн]. Available: <https://unicodes.jesetane.com/>.
- [2] „json“, [Онлайн]. Available: <https://www.json.org/json-en.html>.
- [3] „ECMA-404“, December 2017. [Онлайн]. Available: https://ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf.
- [4] D. Crockford, „rfc4627“, Network Working Group, July 2006. [Онлайн]. Available: <https://www.ietf.org/rfc/rfc4627.txt>.
- [5] „what-is-xml“, Amazon AWS, [Онлайн]. Available: <https://aws.amazon.com/what-is/xml/>.
- [6] „SAML“, webopedia, [Онлайн]. Available: <https://www.webopedia.com/definitions/saml/>.
- [7] K. Lane, „SOAP API“, Postman, 28 June 2023. [Онлайн]. Available: <https://blog.postman.com/soap-api-definition/>.
- [8] „storing-primitives“, stackoverflow, [Онлайн]. Available: <https://stackoverflow.com/questions/50183989/storing-primitives-in-xml>.
- [9] „XSD“, microsoft, [Онлайн]. Available: [https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ms764635\(v=vs.85\)](https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ms764635(v=vs.85)).
- [10] „XSD Schema“, w3schools, [Онлайн]. Available: https://www.w3schools.com/xml/schema_intro.asp.
- [11] „JSON primitives“, yugabyte, [Онлайн]. Available: https://docs.yugabyte.com/preview/api/ysql/datatypes/type_json/primitive-and-compound-data-types/.