|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Architektura zorientowana na usługi** | | | | | | | | | | | |
| 3 | Temat: | ***XML, XSD, XSLT*** | Zadania: | | | | | | | | Data: |
| Autor: | **Sylwia Kaleta** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | **17 X 2018** |
| Autor: | **Kamil Wanat** |  |  |  |  |  |  | - | - | **18:00-19:30** |

**Zadanie 1. Tworzenie XML z klasy**

W tym zadaniu należało w dowolnym języku zaimplementować klasy, wedle specyfikacji podanej w treści zadania podane przez prowadzącego, w naszym wypadku był to język Python:

Zaimplementowane klasy:

W następnym kroku należało w miarę możliwości użyć mechanizmu do serializacji danych w celu stworzenia pliku XML z klasy, niestety jedynym mechanizmem do serializacj danych, jaki udało nam się znaleźć dla języka Python był mechanizm umożliwiający zapis słownika, a nie całego obiektu, dlatego też sami zaimplementowaliśmy potrzebne metody:

Zapis klasy Osoba jako XML:

def toXml(self):  
 if self.typOsoby == TypOsoby.NAUCZYCIEL:  
 typ = "NAUCZYCIEL"  
 else:  
 typ = "UCZEN"  
 xml = "\n".join(["\t\t\t<typOsoby>"+typ+"</typOsoby>",  
 "\t\t\t<imie>" + self.imie + "</imie>",  
 "\t\t\t<nazwisko>" + self.nazwisko + "</nazwisko>",  
 "\t\t\t<rokUrodzenia>" + str(self.rokUrodzenia) + "</rokUrodzenia>"])  
 return xml

Zapis klasy Klasa jako XML:

def toXml(self):  
 xml = "\n".join(["<klasa>",  
 "\t<numer>"+str(self.numer)+"</numer>",  
 "\t<litera>" + str(self.litera) + "</litera>",  
 "\t<numer>" + str(self.numer) + "</numer>",  
 "\t<wychowawca>",  
 self.wychowawca.toXml(),  
 "\t</wychowawca>",  
 "\t<uczniowie>"])  
 for ucz in self.uczniowie:  
 xml = "\n".join([xml,  
 "\t\t<uczen>",  
 ucz.toXml(),  
 "\t\t</uczen>"])  
 xml = "\n".join([xml, "\t</uczniowie>"])  
 return xml

**TUTAJ TRZEBA DODAĆ ODCZYT Z XML!**

**Zadanie 2. Tworzenie XSD z klasy**

W tym zadaniu użyliśmy zewnętrznego narzędzia aby stworzyć plik XSD na podstawie klasy, stworzony plik XSD znajduje się w archiwum z kodem źródłowym pod nazwą klasa.xsd

**Zadanie 3. Walidacja XML**

Celem zadania była walidacja pliku XML na podstawie pliku XSD, w tym celu wykorzystaliśmy bibliotekę xmlschema, która automatycznie przeprowadza walidację pliku:

mySchema = xmlschema.XMLSchema(xsdFilePath)  
print("sprawdzanie poprawnosci:")  
print(mySchema.is\_valid(xmlFilePath))

xsdFilePath – ścieżka do pliku XSD

xmlDilePath – ścieżka do pliku XML

**Zadanie 4. Odczyt i zapis XML**

Zadanie polegało na dodaniu stworzeniu metod, które odpowiednio odczytają lub zapiszą plik XML.

**DODAĆ ODCZYT I WALIDACJĘ DANYCH**

W celu zapisu pliku xml zostały wykorzystane metody z zadania pierwszego.