Serijalizacija metapodataka na primjeru Gettyjevog Tezaurusa za umjetnost i arhitekturu

Kušević, Filip

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2021./2022.

Filip Kušević

Serijalizacija metapodataka na primjeru Gettyjevog Tezaurusa za umjetnost i arhitekturu

Završni rad

Mentor: dr.sc. Goran Zlodi, izv. prof.

Zagreb, rujan 2022.

Izjava o akademskoj čestitosti

Izjavljujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Sadržaj

1. Uvod

U današnje vrijeme digitalizacije i svakodnevnog unaprjeđenja tehnologije pristup raznim informacijama postaje sve jednostavniji. Upravo zbog sve jednostavnijeg pristupa informacijama javlja se i potreba za kvalitetnijim i efikasnijim pronalaskom željenih relevantnih podataka, a osobito u knjižnicama, arhivima, muzejima i sličnim ustanovama. Kako bi se omogućilo pronalaženje i pristup podacima o predmetima baštine, oni moraju sadržavati nekakve opisne karakteristike uz pomoć kojih ih se može pronaći. Sukladno tome, opisuju se metapodacima koji uz opis omogućuju identifikaciju, pristup, prikaz i različite oblike upravljanja. Kako bi se metapodaci učinili otvorenim i iskoristivim i od strane ljudi i od strane strojeva, treba ih moći pretvoriti u povezane otvorene podatke (LOD – *Linked Open Data*). Danas su povezani otvoreni podaci jedna od glavnih poveznica između već spomenutih ustanova, neovisno o kakvoj građi je riječ. Kako bi se osigurala standardizirana vrijednost podataka posebno su važni i tezaurusi kao alati za upravljanje strukturiranim nazivljem koje omogućuje pouzdano označivanje i pronalaženje informacija.

U ovom će radu u prva dva poglavlja ukratko biti opisani metapodaci i povezani otvoreni podaci. U sljedećem poglavlju, detaljno će biti razrađena serijalizacija metapodataka u *Extensible Markup Language*[1] i *JavaScript Object Notation*[2] formatima, za koje je najprije potrebno poznavati *Asynchronous JavaScript and XML*[3] tehnologiju, kao i .NET Framework platformu. Sama serijalizacija u navedenim formatima bit će opisana koristeći *XmlSerializer, DataContractSerializer, DataContractJsonSerializer* te *JavaScriptSerializer* serijalizatore. Nakon toga slijedi usporedba XML i JSON formata te neki od načina pretvaranja između XML-a i JSON-a, a zatim je opisan i JSON-LD format koji u usporedi sa standardnim JSON-om ima nekoliko određenih prednosti, a jedna od njih uključuje i efikasno povezivanje podataka. Iduće poglavlje fokusira se na Gettyjev Tezaurus za umjetnost i arhitekturu[4] kao najpoznatiji tezaurus za područje umjetnosti i arhitekture.

[1] U daljnjem tekstu „XML“.
[2] U daljnjem tekstu “JSON”.
[3] U daljnjem tekstu “AJAX”.
[4] U daljnjem tekstu „AAT“.

U tom pogljavlju bit će obuhvaćena povijest AAT-a, kao i njegov opseg i struktura koji uključuju fasete i hijerarhije, a zatim i poveznice AAT-a s otvorenim povezanim podacima te JSON-LD formatom.

2. Metapodaci

Jedno od najosnovnijih objašnjenja metapodataka jest da su oni „podaci o podacima“ (Lubas, Jackson i Schneider, 2013, p. 2). Iako navedena definicija ne govori puno o njima, oni su upravo to, dodatni podaci. Metapodaci se, u najširem smislu, koriste za opisivanje resursa informacija, što bi značilo da se koriste na internetu, kao i u knjižnicima, arhivima i drugim ustanovama. Oni objašnjavaju dani izvor ili informaciju, odnosno govore o njegovom sadržaju, autora, datumu nastanka, pravima korištenja itd. Metapodaci su od velike važnosti kada je riječ o ne-tekstualnim informacijskim izvorima poput fotografija ili audio-sadržaja i video-sadržaja. Takve vrste izvora ne mogu ponuditi podatke u tekstualnom obliku, koji bi bio puno lakši za pretraživanje, pa su zato potrebni metapodaci kao set podataka koji će pomoći u pretraživanju i korištenju takvih vrsta izvora. Čak i kada se ne radi o ne-tekstualnim izvorima, koriste se metapodaci. No, kako su područja koja su počela koristiti metapodatke i stvarati svoje kataloge rasla, tako je rasla i potreba za drugim standardima, koji bi se prilagodili određenom području. Zbog toga su nastali standardi posebno za arhive ili muzeje, kao i za određenu vrstu građe (npr. fotografije). Spomenuti rast nije stao samo na tome. Već 1990-ih godina s pojavom interneta i tehnoloških inovacija pojavila se i mogućnost interoperabilnosti – „sposobnosti više sustava s različitim hardverskim i softverskim platformama, podatkovnim strukturama i sučeljima za razmjenu podataka uz minimalan gubitak sadržaja i funkcionalnosti“ (NISO, 2004, p. 2). Metapodaci i interoperabilnost imaju veliku ulogu u ustanovama poput knjižnica i arhiva zbog lakšeg pronalaženja, snalaženja i međusobne komunikacije između ustanova, ali i korisnika.

3. Povezani otvoreni podaci

Povezane otvorene podatke najjednostavnije je razumijeti tako da se najprije napravi podjela na otvorene podatke (engl. *open data)* i povezane podatke (engl. *linked data)*. „Otvoreni podaci su podaci koji su slobodno dostupni i mogu ih koristiti i ponovno objavljivati svi bez ograničenja autorskih prava ili patenata“ (Braunschweig et al, 2012, p. 1). S druge strane, „povezani podaci odnose se na podatke objavljene na webu na takav način da su strojno čitljivi, da im je značenje eksplicitno definirano, da su povezani s drugim vanjskim skupovima podataka, a zauzvrat se mogu

povezati i iz vanjskih skupova podataka“ (Yu, 2011, p. 409). Sukladno tome, može se zaključiti da su povezani otvoreni podaci zapravo povezani podaci za višekratnu besplatnu upotrebu koji su objavljeni i dostupni svima, odnosno pod otvorenom licencom (Berners-Lee, 2006).

Naravno, svi podaci, pa tako i povezani otvoreni podaci, variraju u kvaliteti. Upravo iz tog razloga, Berners-Lee (2006) razvio je tzv. sustav ocjenjivanja zvjezdicama kako bi se odredila kvaliteta podataka prema sljedećim kategorijama :

★ - Dostupni na webu (bez obzira na format), ali s otvorenom licencom, da budu otvoreni podaci

★★ - Dostupni kao strojno čitljivi strukturirani podaci (npr. Microsoft Excel umjesto skenirane slike tablice)

★★★ - Sve od navedenog, ali koristi se nevlasnički format (npr. vrijednosti odvojene zarezom[5] umjesto Microsoft Excel-a)

★★★★ - Sve od navedenog, ali koriste se otvoreni standardi iz *World Wide Web Consortium*-a, RDF i SPARQL, za prepoznavanje stvari. „RDF (*Resource Description Framework*) formalni je jezik koji definira osnovnu strukturu povezanih podataka koji čine semantički web“ (Coyle, 2012, p. 11), a „SPARQL je standardni jezik za postavljanje upita RDF podacima“ (Pérez, Arenas i Gutierrez, 2009, p. 1). Navedeni standardi koriste se kako bi ljudi mogli ukazati na pojedinačne podatke

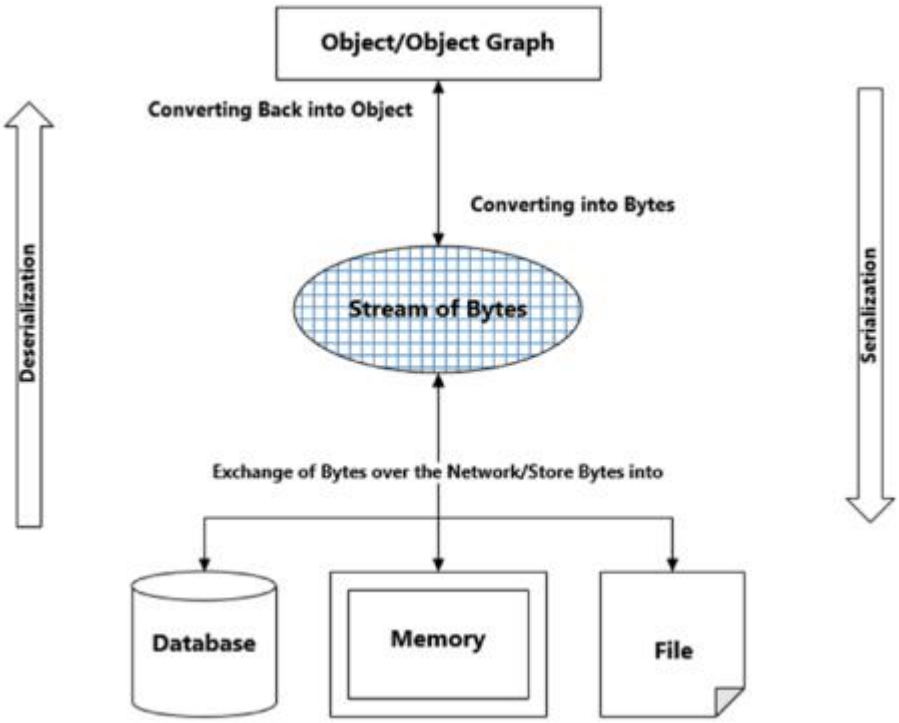
★★★★★ - Sve od navedenog, ali potrebno je povezati svoje podatke s podacima drugih ljudi kako bi se pružio kontekst.

Također, Berners-Lee (2006) navodi kako je moguće imati povezane podatke s pet zvjezdica bez da su oni otvoreni, ali ako se za neke podatke tvrdi da su povezani otvoreni podaci, oni moraju biti otvoreni kako bi uopće mogli dobiti ikakvu zvjezdicu. Navedeni sustav ocjenjivanja zvjezdicama razvijen je s primarnim ciljem poticanja ljudi da svoje podatke dobro povežu, a time i omoguće ostalim ljudima što jednostavnije korištenje tih podataka.

[5] U daljnjem teksu „CSV“ (engl. *Comma-separated values*)

4. Serijalizacija metapodataka

Međusobna povezanost, odnosno interoperabilnost različitih baza podataka u fokusu je velikog broja istraživanja provedenih kroz već duži niz godina u određenim zajednicama, ali primarno u zajednici baze podataka. Posljedično, želja za sve lakšim pristupom podacima, novi načini pohrane podataka, kao i razmjena podataka između različih repozitorija podataka postala je još važnija. Razmjena podataka, kao i pohranjivanje podataka u takvim različitim sustavima zahtijeva određeni neutralni format podataka koji je neovisan o jeziku ili platformi te koji svi sustavi imaju mogućnost razumijeti. Upravo zbog toga je važan proces serijalizacije, odnosno „proces pretvaranja objekta ili grafa objekta u tok bajtova. To je proces pretvaranja objekta u bajtove ili tekst kako bi se pohranio u bilo koju vrstu pohrane ili razmijenio objekt preko mreže." (Asad i Ali, 2017, p. 305). S druge strane, jednostavno govoreći, deserijalizacija je samo obrnuti proces, odnosno pretvaranje toka bajtova u objekt ili graf objekta. „Serijalizirani objekt nosi podatke objekta u obliku toka zajedno s informacijama o tipu objekta, odnosno njegovoj verziji, kulturi i nazivu sklopa." (Asad i Ali, 2017, p. 306). Procesi serijalizacije i deserijalizacije prikazani su i uz pomoć Slike 2. kao što su to prikazali Asad i Ali (2017, p. 306):



Slika 1. Serijalizacija i deserijalizacija. Asad, A. i Ali, H. (2017) *The C# Programmer's Study Guide (MCSD) Exam: 70-483*. 1st edn. Berkeley: Apress.

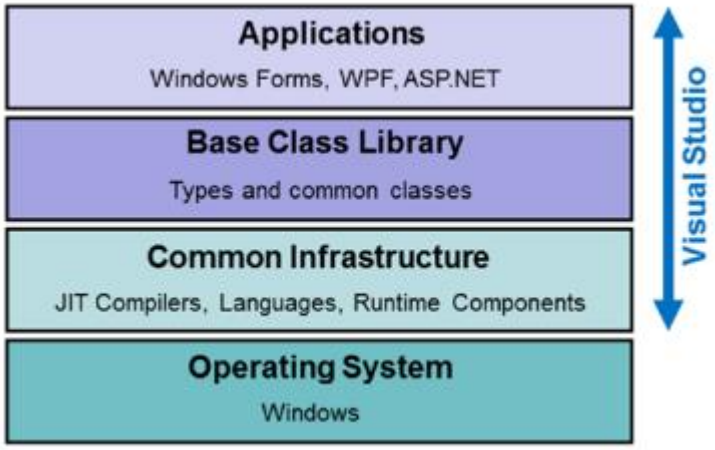
Serijalizacija i deserijalizacija pružaju mogućnost razmjene podataka u standardiziranom formatu. Na taj način različite aplikacije imaju mogućnost međusobne komunikacije i dijeljenja znanja, čak i u slučaju razlike u modelima podataka koji su inače specifični za svaku aplikaciju. Postoji više formata serijalizacije, ali poglavlja u nastavku fokusirat će se isključivo na XML i JSON formate kao jedne od najpoznatijih i najčešće korištenih formata serijalizacije.

4.1. AJAX

AJAX jedan je od veoma značajnih razvoja na webu koji web aplikacijama omogućuje da izgledaju kao aplikacije za radnu površinu (engl. *desktop application*). Obično, u slučajevima kada je na normalnim web stranicama bilo potrebe za dodatnim informacijama, bilo je nužno ponovno učitati cijelu stranicu. Međutim, uz pomoć AJAX-a, postalo je moguće izvršiti prijenos informacija bez ponovnog učitavanja ili osvježavanja stranice. (Haq, Faraz Khan i Hussain, 2013). U praksi, navedeno se može jasno utvrditi koristeći primjerice Google pretraživanje nekog proizvoljnog pojma. Prilikom upisivanja određenog pojma, Google će automatski ponuditi niz prijedloga koje bi na temelju upisanog teksta korisnik potencijalno želio pretražiti. Navedeni primjer potvrđuje da AJAX tehnika omogućuje stranicama prijenos male količine podataka, kao i dohvaćanje podataka s poslužitelja dok se korisnik nalazi na istoj stranici, bez ponovnog učitavanja cijele stranice. Za prenošenje takvih podataka, potreban je format za prijenost podataka, od kojih su najkorišteniji XML ili JSON formati.

4.2. .NET Framework

„Microsoft-ov .NET Framework platforma je za izgradnju aplikacija, komponenti i usluga temeljenih na Windowsu i webu.“ (Joshi, 2017, p. 11). Joshi (2017, p. 11) jednostavno je prikazao vrstu podatkovne strukture čija svrha je pohrana istovrsnih elemenata, odnosno stog (engl. *stack*) .NET Framework-a:



Slika 2. Stog .NET Framework-a. Joshi, B. (2017). *Beginning XML with C# 7*. 2nd edn. Berkeley: Apress.

Na najnižem sloju nalazi se operativni sustav (engl. *Operating System*). U pogledu .NET Framework-a, podrazumijeva se jedna od različitih verzija Windows operativnih sustava, poput Windows 7, Windows 8.x ili Windows 10. Iznad sloja operativnog sustava, nalazi se sloj zajedničke infrastrukture (engl. *Common Infrastructure*) koji se sastoji od Just-In-Time[6] kompilatora (engl. *compiler*), jezika i ostalih komponenti vremena izvođenja (engl. *runtime components*). Navedeni sloj omogućava izvršno okruženje, kao i usluge vremena izvođenja (engl. *runtime services*), a ovom sloju dodijeljene su i određene odgovornosti, neke od njih podrazumijevaju JIT kompilaciju, upravljanje memorijom, upravljanje nitima te sigurnosnu provjeru (Joshi, 2017). Iznad sloja zajedničke infrastrukture instalira se kolekcija klasa velikog razmjera koja se naziva knjižnica osnovnih klasa (engl. *Base Class Library*), a ista obuhvaća klase za ulaz i izlaz datoteka, pristup bazi podataka, XML manipulaciju, web programiranje i velik broj drugih stvari (Joshi, 2017). Knjižnica osnovnih klasa u suštini pruža klase za izvođenje više-manje svih potrebnih radnji u određenoj aplikaciji. Razvijajući neku aplikaciju u .NET Framework-u, vrlo vjerojatno će se pojaviti potreba za korištenjem barem neke klase koja je dostupna u knjižnici osnovnih klasa. U konačnici, najviši sloj sastoji se od različitih modela primjene, a neki od izbora su *Windows Forms*, *Windows Presentation Foundation* i ASP.NET te je za razvoj navedenih aplikacija potrebno koristiti *Visual Studio*, što znači da se pisanje koda, kompiliranje projekta, otklanjanje pogrešaka koda i zadaci sličnog

[6] U daljnjem tekstu “JIT”.

karaktera odvijaju unutar integriranog razvojnog okruženja (engl. *Integrated development environment*), odnosno *Visual Studio* IDE-a (Joshi, 2017).

Nadalje, već spomenute klase .NET Framework-a imaju ulogu pomoći prilikom serijalizacije i deserijalizacije objekta, a .NET Framework također nudi različite načine za konfiguraciju vlastitih objekata. Standardno postoje 3 mehanizma serijalizacije, odnosno deserijalizacije koje nudi .NET Framework (Asad i Ali, 2017):

1. *BinaryFormatter*
2. *XmlSerializer*
3. *DataContractSerializer*

BinaryFormatter serijalizator je za serijalizaciju i deserijalizaciju podataka u binarnom formatu te je stroži, pa tako i sigurniji od ostalih serijalizatora. *XmlSerializer* koristi se za serijalizaciju i deserijalizaciju objekta u XML dokumentu te omogućuje kontrolu načina kodiranja objekata u XML. Naposljetku, Asad i Ali (2017) navode da se *DataContractSerializer* također koristi za serijalizaciju objekta u XML tok (engl. *stream*), ali korištenjem isporučenog ugovora o podacima (engl. *supplied data contract*). Osim navedenih serijalizatora, postoje i neki drugi koji se koriste za serijalizaciju i deserijalizaciju podataka (Asad i Ali, 2017):

1. *DataContractJsonSerializer*
2. *JavaScriptSerializer*

DataContractJsonSerializer serijalizira objekte u JSON i deserijalizira JSON podatke u objekte, dok *JavaScriptSerializer* serijalizira i deserijalizira objekte za aplikacije s omogućenim AJAX-om (Asad i Ali, 2017).

4.3. XML oblik serijalizacije

XML jezik je za označavanje, kao i jednostavan standard koji se može koristiti za opisivanje, transformaciju i kodiranje teksta, kao i podataka te se može obrađivati i transformirati preko više različitih platformi. „XML je odobren kao preporuka od strane *World Wide Web Consortiuma* u veljači 1998. godine.“ (Joshi, 2017, p. 1). XML podaci su metapodaci, a XML nudi i standardiziran način za predstavljanje tekstualnih podataka. Navedeni XML podaci često se nazivaju i XML dokumentom. U korištenju XML-a, prema propisanoj gramatici, svi podaci pohranjuju se u

oznakama koje moraju započeti s odabranom oznakom otvorenog oblika, zatim se upisuju željeni podaci te završavaju s istom oznakom, ali zatvorenog oblika. Specifikacija XML jezika osigurava da oblik serijaliziranih podataka XML-a ima dobru sigurnost. U isto vrijeme, „skalabilnost oznaka XML jezika omogućuje programerima konfiguraciju ovisno o karakteristikama aplikacija web servisa, tako da se aplikacije web servisa mogu prilagoditi različitim situacijama.“ (Wang, 2011, p. 182).

Samostalni XML podaci ne vrše nikakve radnje. Sukladno tome, kako bi se ti podaci uspješno obradili, potrebno je koristiti dio softvera koji se naziva raščlanjivač (engl. *parser*). Također, „za razliku od jezika za označavanje hiperteksta (HTML - *Hypertext Markup Language*), koji se fokusira na to kako prezentirati podatke, XML se fokusira na to kako predstaviti podatke.“ (Joshi, 2017, p. 1). Isto tako, XML je jednostavniji i lakši za upravljanje u usporedbi sa standardnim generaliziranim jezikom za označavanje (SGML – *Standard Generalized Markup Language*). XML nije u potpunosti neovisan jezik za označavanje, već je alat koji korisnicima omogućava definiranje vlastitog jezika za označavanje. Sukladno, XML se sastoji od korisnički definiranih oznaka, odnosno to znači da svaki korisnik može slobodno definirati i koristiti vlastite oznake u XML dokumentu. S obzirom na to da su te oznake definirane od strane korisnika, puno su jednostavnije i čitljivije kako strojevima, tako i drugim korisnicima.

Nadalje, s obzirom na to da je XML preporuka *World Wide Web Consortiuma*, isti ima brojne koristi. Upravo zbog te preporuke, XML je zaprimio je status industrijskog standarda koji mu je pomogao da bude široko prihvaćen. Također, „XML dokumenti su samoopisni te su zbog oznaka za označavanje lakše čitljivi od primjerice datoteka s vrijednostima odvojenim zarezom (CSV).“ (Joshi, 2017, p. 2). Isto tako, XML je proširiv (engl. e*xtensible*). Uz navedeno, s obzirom na to da XML dokumenti upotrebljavaju oznake za označavanje, XML se može i jednostavno obraditi uz pomoć raščlanjivača koji s lakoćom može pročitati te oznake (Joshi, 2017). Bitno je za istaknuti da je CSV format tradicionalno bio standardni format predstavljanja i transfera podataka te da je za obradu podataka te vrste nužno poznavati točnu poziciju zareza (,) ili nekog drugog određenog graničnika u upotrebi, što je uvelike

otežavalo čitanje i pisanje dokumenata. Iz navedenog su potencijalno proizlazili ozbiljniji problemi, ako se primjerice u tom trenutku radi o nizu potpuno različitih i nepoznatih CSV datoteka (Joshi, 2017). XML se može koristiti i za jednostavnu razmjenu i dijeljenje podataka, a u konačnici i za stvaranje specijaliziranih rječnika. Koristeći XML kao bazu, moguće je kreirati vlastite rječnike. „Protokol bežične aplikacije (WAP – *Wireless Application Protocol*), jezik za označavanje bežične veze (WML – *Wireless Markup Language*) i jednostavni protokol za pristup objektu (SOAP – *Simple Object Access Protocol*) neki su od primjera za specijalizirane XML rječnike.“ (Joshi, 2017, p. 3).

U nastavku se nalazi primjer jednostavnog XML dokumenta kako je to naveo Wang (2011, p. 183):

```
<?xml version="1.0" encoding="UTF-8"?>
<contacts>
        <contact>
                <name>
                        Jack Sparrow
                </name>
                <phone>
                        123-532-5392
                </phone>
                <work>
                        actor
                </work>
        </contact>
        <contact>
                ...
        </contact>
        ...
</contacts>
```

Bitno je za napomenuti da XML sintaktsa dopušta postojanje dviju oznaka koje imaju isti naziv na istoj razini, navedeno se može zaključiti iz prehodnog primjera oznake „<contact>“. XML u praksi može sadržavati takve oznake, a oznake s istim nazivom

također su i standardna praksa u HTML jeziku. Međutim, u slučajevima kada se za transfer podataka koristi XML serijalizacija, postoji velika mogućnost da će oznake s istim nazivima uzrokovati poteškoće u indeksiranju podataka kod primatelja (Wang, 2011).

4.3.1. XmlSerializer

XML serijalizacija omogućuje serijalizaciju objekta u XML format ili XML tok, a XML objekti analiziraju se kao DOM-ovi (*Document Object Model*). Potrebno je istaknuti da se u XML serijalizaciji mogu serijalizirati isključivo javna polja ili svojstva te ne uključuje podatke o tipu (engl. *Type*) serijaliziranog objekta. Za primjer se može uzeti serijalizirani objekt tipa „Roditelj“ za koji se, nakon što se isti deserijalizira, ne može sa stopostotnom sigurnošću tvrditi da će biti deserijaliziran u objekt tipa „Roditelj“. Upravo iz tog razloga, XML serijalizacija ne pohranjuje informacije o tipu objekta, a također ni ne pretvara metode, indeksatore, privatna polja ili svojstva samo za čitanje (osim kolekcija samo za čitanje), već je za serijalizaciju svih polja i svojstava objekta, javnih i privatnih, potrebno upotrijebiti *DataContractSerializer* umjesto XML serijalizacije (Asad i Ali, 2017).

Dakle, XML serijalizacija koristi klasu *XmlSerializer* za samu implementaciju XML serijalizacije. Ranije je bilo navedeno kako je od spomenutih serijalizera najstroži *BinaryFormatter*. *XmlSerializer* nije strog kao *BinaryFormatter*, ali isto tako njegov učinak nije najbolji te ne održava informacije o objektu, a također nije moguće ni serijalizirati privatna polja (Asad i Ali, 2017). Kako bi se XML serijalizacija izvršila, potrebno je tip označiti atributom *Serializable* koji .NET Framework-u poručuje da je taj tip potrebno serijalizirati. Navedeni atribut provjerit će objekt u pitanju, kao i graf objekata, odnosno sve objekte na koje se poziva, kako bi se osigurala serijalizacija svih povezanih objekata (Asad i Ali, 2017). Primjer u nastavku pokazuje kako se može konfigurirati objekt za XML serijalizaciju, zatim njegovu serijalizaciju u datoteku i deserijalizaciju u objekt uz pomoć *XmlSerializer* klase (Asad i Ali, 2017):

```
[Serializable]
public class Teacher
{
        public int ID { get; set; }
```

```
        public string Name { get; set; }
        public long Salary { get; set; }
}
XmlSerializer xml = new XmlSerializer(typeof(Teacher));
using (var stream = new FileStream("Sample.xml", FileMode.Create))
{
        xml.Serialize(stream, t);
}

Console.WriteLine("Data has been Serialized!");

Teacher teacher = null;
using (var stream = new FileStream("Sample.xml", FileMode.Open))
{
        XmlSerializer xml = new XmlSerializer(typeof(Teacher));
        teacher = (Teacher)xml.Deserialize(stream);
}

Console.WriteLine(teacher.ID);
Console.WriteLine(teacher.Name);
Console.WriteLine(teacher.Salary);

Console.WriteLine("Data has been Deserialized!");
```

Serijalizirani objekt na ovaj način u XML formatu izgleda ovako (Asad i Ali, 2017):

```
<?xml version="1.0"?>
<Teacher xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <ID>2</ID>
        <Name>Ahsan</Name>
        <Salary>20000</Salary>
</Teacher>
```

XML serijalizacija može se i konfigurirati kako bi se postigla veća kontrola nad tipom koji će biti serijaliziran tako da se koriste atributi koje omogućuje imenski prostor (engl. *namespace*) *System.Xml.Serialization*, a neki od važnijih atributa, zajedno s njihovom upotrebom, koji su u standardnoj upotrebi su (Asad i Ali, 2017):

1. *XmlRoot* – primjenjuje se na tip, što kompilatoru govori da će ovo biti glavni, odnosno roditeljski čvor serijaliziranog objekta u XML-u.
2. *XmlAttribute* – primjenjuje se na bilo koje od javnih polja mapiranih u atribut na njegovom roditeljskom čvoru.
3. *XmlElement* – primjenjuje se na bilo koje od javnih polja mapiranih u element roditeljskog čvora.
4. *XmlIgnore* – primjenjuje se na bilo koje javno polje koje neće biti serijalizirano.
5. *XmlArray, XmlArrayItem* – primjenjuju se na bilo koje od javnih polja kolekcije tipova za serijalizaciju.

Standardno je svako javno polje tipa serijalizirano kao *XmlElement*, a korištenje navedenih atributa omogućava mapiranje objekta u odgovarajući XML format. U nastavku slijedi primjer kako dodatno konfigurirati tip za XML serijalizaciju (Asad i Ali, 2017):

```
[Serializable]
[XmlRoot("Teacher")]
public class teacherClass
{
        [XmlAttribute("ID")]
        public int id { get; set; }
        [XmlElement("Name")]
        public string name { get; set; }
        [XmlIgnore]
        public long salary { get; set; }
        [XmlElement("Students")]
        public studentClass st { get; set; }
}
[Serializable]
public class studentClass
{
```

```
        [XmlAttribute("RollNo")]
        public int rollno { get; set; }
        [XmlElement("Marks")]
        public int marks { get; set; }
}
//Serialization
teacherClass t = new teacherClass
{
        id = 2,
        name = "Ahsan",
        salary = 20000,
        st = new studentClass
        {
                rollno = 1,
                marks = 50
        }
};
XmlSerializer xml = new XmlSerializer(typeof(teacherClass));
using (var stream = new FileStream("Sample.xml", FileMode.Create))
{
        xml.Serialize(stream, t);
}
Console.WriteLine("Data has been Serialized!");

//Deserialization
teacherClass teacher = null;
using (var stream = new FileStream("Sample.xml", FileMode.Open))
{
        XmlSerializer xml = new XmlSerializer(typeof(teacherClass));
        teacher = (teacherClass)xml.Deserialize(stream);
}

Console.WriteLine(teacher.id);
Console.WriteLine(teacher.name);
Console.WriteLine(teacher.salary);
```

```
Console.WriteLine(teacher.st.rollno);
Console.WriteLine(teacher.st.marks);
Console.WriteLine("Data has been Deserialized!");
```

Serijalizirani objekt ovog primjera u XML formatu izgleda ovako (Asad i Ali, 2017):

```
<?xml version="1.0"?>
<Teacher xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" ID="2">
 <Name>Ahsan</Name>
 <Students RollNo="1">
        <Marks>50</Marks>
 </Students>
</Teacher>
```

Navedeni primjer prikazuje serijalizaciju objekta *teacherClass*, kao i svih ostalih objekata povezanih s tim objektom, odnosno graf objekta. Još jednom bitno je za istaknuti da tip mora biti javan za XML serijalizaciju, jer *XmlSerializer* serijalizira isključivo javne tipove ili članove.

4.3.2. DataContractSerializer

Prethodno je već spomenuto da *DataContractSerializer* serijalizira objekt u XML format koristeći isporučeni ugovor o podacima (Asad i Ali, 2017). *Windows Communication Foundation*[7] okvir je za izgradnju servisno orijentiranih aplikacija te je važno za istaknuti da WCF koristi *DataContractSerializer* kao standardni zadani serijalizator, a u radu s WCF-om tipovi se serijaliziraju tako da mogu biti poslani ostalim aplikacijama (Asid i Ali, 2017). Kao što to navode Asid i Ali (2017), glavne razlike između *DataContractSerializer* i *XmlSerializer* serijalizatora su:

1. Umjesto korištenja *Serializable* atributa, koristi se atribut *DataContract*.
2. Članovi nisu standardno serijalizirani, kao što je to u *XmlSerializer* serijalizatoru.
3. Svi članovi koje se želi serijalizirati moraju biti eksplicitno označeni s atributom *DataMember*.

[7] U daljnjem tekstu “WCF”.

4. Kako bi se određeni član zanemario u serijalizaciji, potrebno je koristiti atribut *IgnoreDataMember* umjesto *XmlIgnore*.
5. *DataContractSerializer* također može serijalizirati i privatna polja, što nije moguće u *XmlSerializer* serijalizatoru.
6. U *DataContractSerializer* serijalizatoru potrebno je za serijalizaciju objekta koristiti metodu *WriteObject()* te metodu *ReadObject()* za deserijalizaciju toka u objekt.

U nastavku se nalazi primjer serijalizacije i deserijalizacije objekta uz pomoć *DataContractSerializer* serijalizatora (Asad i Ali, 2017):

```
[DataContract]
public class Teacher
{
        [DataMember]
        private int id = 1;
        [DataMember]
        public string name { get; set; }
        [IgnoreDataMember]
        public long salary { get; set; }
}
//Serialization
DataContractSerializer dataContract = new DataContractSerializer(typeof(Teacher));
using (var stream = new FileStream("Sample.xml", FileMode.Create))
{
        dataContract.WriteObject(stream, t);
}

Console.WriteLine("Data has been Serialized!");

//Deserialization
Teacher teacher = null;
DataContractSerializer dataContract = new DataContractSerializer(typeof(Teacher));

using (var stream = new FileStream("Sample.xml", FileMode.Open))
{
```

```
        teacher = (Teacher)dataContract.ReadObject(stream);
}

Console.WriteLine("Data has been Deserialized!");
```

DataContractSerializer također je moguće koristiti iz imenskog prostora *System.Runtime.Serialization* jednako kao što se koristio i *XmlSerializer*, uzimajući u obzir da dolazi do razlike u atributima ili metodama za serijalizaciju i deserijalizaciju.

4.4. JSON oblik serijalizacije

JSON je lagani, tekstualni format za razmjenu podataka neovisan o jeziku za serijalizaciju strukturiranih podataka. U posljednjih nekoliko godina, JSON je stekao ogromnu popularnost među web programerima i postao je glavni format za razmjenu informacija putem weba, a danas igra ključnu ulogu u web aplikacijama, što se i očituje u njegovoj popularnosti za slanje API (*Application Programing Interaface*) zahtjeva i odgovora preko HTTP protokola (Pezoa et al, 2016). JSON dobiva sve veću popularnost u odnosu na XML zbog svoje jednostavnosti, kompaktnosti i mogućnosti uklapanja u objektne tipove programskih jezika. JSON definira mali skup pravila oblikovanja za prijenosno predstavljanje strukturiranih podataka i može predstavljati četiri primitivna tipa (nizovi znakova, brojevi, logičke vrijednosti i null) i dva strukturirana tipa (objekti i nizovi), a izveden je iz literala objekta JavaScripta, kako je definirano u standardu programskog jezika ECMAScript (Bray, 2014). Mnogi sustavi složili su se koristiti JSON za prijenos podataka. Međutim, postoje formati za razmjenu podatka poput XML-a koji su bili tu prije nego što se o JSON-u uopće započelo razmišljati, pa tako postoje i mnogi sustavi koji još uvijek koriste primjerice CSV format (Basset, 2015). Kao što je već spomenuto, JSON je lagani format za razmjenu podataka koji zbog toga ljudima olakšava čitanje i pisanje, a strojevima lagano raščlanjivanje i generiranje. Navedena svojstva JSON čine idealnim za razmjenu podataka.

Dodatno, od dva strukturirana tipa u JSON-u, objekti i nizovi (engl. *array*), objekti su parovi imena i vrijednosti, a nizovi su uređene liste vrijednosti. Parovi imena i vrijednosti u JSON-u realizirani su kao objekti, ali u raznim jezicima mogu se

realizirati kao objekti, zapisi, rječnici, popisi s ključevima i sl. S druge strane, uređene liste vrijednosti u većini ostalih jezika mogu biti realizirane kao nizovi, vektori, liste ili kao sekvence. Navedene strukture podataka su univerzalne, što znači da ih svi moderni programski jezici podržavaju u nekom obliku. Upravo zbog toga, smisleno je da se format podataka koji je razmjenjiv između programskih jezika temelji na ovim strukturama. U nastavku se nalazi jednostavan prikaz JSON formata (Wang, 2011, p. 183):

```
{
        "contacts": {
                "contact": {
                        "name": "Jack Sparrow",
                        "phone": "123-532-5392"
                        "work": "actor"
                }
        }
}
```

4.4.1. DataContractJsonSerializer

JSON serijalizacija serijalizira objekt u JSON format, vrlo učinkovit format kodiranja koji je posebice koristan prilikom slanja manjih količina podataka između klijenata (preglednika) i web usluga koje imaju omogućen AJAX, a JSON serijalizacijom automatski upravlja WCF kada se koriste *DataContract* tipovi u servisnim operacijama koje su izložene preko krajnjih točaka omogućenih za AJAX (Asad i Ali, 2017).

DataContractJsonSerializer služi za pretvaranje objekta u JSON podatke, kao i za ponovno pretvaranje JSON podataka u objekt. *DataContractJsonSerializer* klasa je koju pruža .NET Framework u imenskom prostoru *System.Runtime.Serialization.Json* te podržava iste tipove kao i *DataContractSerializer*, a jednako tako pruža metodu *WriteObject()* za serijalizaciju i metodu *ReadObject()* za deserijalizaciju (Asad i Ali, 2017). Ostatak JSON postupka serijalizacije isti je kao i kod ostalih serijalizatora te se uglavnom koristi s WCF-om. Potrebno je istaknuti da su privatni članovi također serijalizirani u JSON serijalizaciji,

a prikaz JSON serijalizacije i deserijalizacije uz pomoć *DataContractJsonSerializer* serijalizatora nalazi se u nastavku (Asad i Ali, 2017):

```
[DataContract]
public class Teacher
{
	[DataMember]
	private int id = 1;
	[DataMember]
	public string name { get; set; }
	[DataMember]
	public long salary { get; set; }
}

//Serialization
DataContractJsonSerializer dataContract = new DataContractJsonSerializer(typeof(Teacher));
	using (var stream = new FileStream("Sample.json", FileMode.Create))
	{
		dataContract.WriteObject(stream, t);
	}
	Console.WriteLine("Data has been Serialized!");

//Deserialization
Teacher teacher = null;
DataContractJsonSerializer dataContract = new DataContractJsonSerializer(typeof(Teacher));
	using (var stream = new FileStream("Sample.json", FileMode.Open))
	{
		teacher = (Teacher)dataContract.ReadObject(stream);
	}
	Console.WriteLine("Data has been Deserialized!");
```

Deserijalizirani objekt na temelju ovog prikaza izgleda ovako:

```
{ "id":1,"name":"Ahsan","salary":20000}
```

4.4.2. JavaScriptSerializer

JavaScriptSerializer klasa je koju pruža .NET Framework u imenskom prostoru *System.Web.Script.Serialization* koji se nalazi u *System.Web.Extension* sklopu te koji se koristi za serijalizaciju i deserijalizaciju objekta u JSON format za aplikacije koje imaju omogućen AJAX, a u nastavku je naveden osnovni način serijalizacije i deserijalizacije objekta koristeći *JavaScriptSerializer* (Asad i Ali, 2017):

```
private class Teacher
{
        private int id { get; set; }
        public string name { get; set; }
        public long salary { get; set; }
}

//Serialization
JavaScriptSerializer dataContract = new JavaScriptSerializer();
string serializedDataInStringFormat = dataContract.Serialize(steacher);

Console.WriteLine("Data has been Serialized!");

//Deserialization
Teacher dteacher = null;
dteacher = dataContract.Deserialize<Teacher>(serializedDataInStringFormat);

Console.WriteLine("Data has been Deserialized!");
```

4.5. Usporedba XML-a i JSON-a

Jedna od razlika XML-a i JSON-a može se uočiti na ranijim primjerima jednostavnih XML i JSON formata koje je prikazao Wang (2011), a to je postojanje više jednakih oznaka u dokumentu. XML dozvoljava takve oznake, dok se nakon prijevoda iz XML-a u JSON takve višestruke jednake oznake moraju biti uklonjene. Povijesno gledano, XML je imao brojne prednosti nad JSON-om dok je JSON još bio novitet. Tadašnje

prednosti XML-a uključivale su vrlo jaku podršku za imenske prostore, shemu koja je imala više mogućnosti proširivanja, dugotrajno postojanje na tržištu što je rezultiralo podrškom od strane većine razvojnih alata, visokom razinom podrške za proizvode povezane s web uslugama i brojne druge (Haq, Faraz Khan i Hussain, 2013). Međutim, u današnje vrijeme JSON je prerastao XML u popularnosti, a ta popularnost još uvijek kontinuirano raste svakim danom. Popularnost JSON-a rezultat je njegovih brojnih prednosti, neke od njih uključuju podršku većine različitih preglednika, njegov format vrlo je sažet zbog pristupa koji se temelji na ime/vrijednost paru, ima dobru podršku od većine *JavaScript* knjižnica i AJAX alata, ima jednostavan API za *JavaScript* i brojne druge jezike itd. (Haq, Faraz Khan i Hussain, 2013).

U trenutku prve pojave i promocije AJAX tehnologije, XML jezik bio je korišten u širokom rasponu, ali kako je AJAX tehnologija postajala popularnija, sve više su se primjećivali nedostaci XML-a kod primjene na interaktivnu stranicu (Wang, 2011). Iako XML ima adekvatne specifikacije, njegova struktura je prenapuhana, što uzrokuje nisku učinkovitost njegove analize, a isto tako XML dokumenti analiziraju se kao DOM-ovi (*Document Object Mode*), što zahtijeva puno više vremena, a ako se od strane programera intenzivno koristi XML u laganim AJAX aplikacijama za razmjenu podataka, učinkovitost same stranice bit će uvelike smanjena (Wang, 2011). Iako je XML sigurniji, JSON ima puno veću učinkovitost parsiranja. S obzirom na to da lagane AJAX aplikacije imaju manje zahtjeve u pogledu sigurnosti, dok su zahtjevi za učinkovitošću poprilično visoki te imaju dobru podršku za *JavaScript*, JSON ima ogromnu primjenu u usporedbi s XML-om (Wang, 2011). XML je zapravo idealan izbor za prijenos dokumenata koji sadrže velik broj različitih vrsta podataka i elemenata, ali JSON je prikladniji za dinamičke web aplikacije i jednostavne prijenose podataka, a takve aplikacije danas su sve češće.

4.6. Pretvaranje između XML-a i JSON-a

Razlićiti programski jezici posjeduju različite načine pretvarnja između XML-a i JSON-a. Primjerice, u Java programskom jeziku, uz pomoć klase *org.json.XML* mogu se jednostavno pretvoriti JSON podaci u XML podatke, odnosno XML podaci u JSON podatke. XML se može pretvoriti u JSON tako da se koristi metoda te klase

XML.toJSONObject. Dok se za obrnuto pretvaranje, odnosno iz JSON-a u XML, koristi metoda *XML.toString()*.

Osim Jave, Python također sadrži određene načine za pretvaranje. U Python-u se XML podaci mogu pretvoriti u JSON podatke uz pomoć *xmltodict* i JSON modula. *Json* je ugrađeni Python modul te nema potrebe za ugradnjom. Funkcija *xmltodict.parse()* pretvara XML podatke u Python rječnik, a zatim funkcija *json.dumps()* uzima te pretvorene podatke u obliku rječnika te ih dalje pretvara u JSON format. S druge strane, pretvaranje podataka iz JSON-a u XML u Pythonu moguće je tako da se JSON podaci najprije učitaju uz pomoć metode *json.loads()*, a zatim se upotrijebi *dicttoxml* modul kako bi se JSON pretvorio izravno u XML. Osim navedenih načina pretvaranja u spomenutim programskim jezicima i mnogi ostali programski jezici također sadrže vlastite načine pretvaranja podataka između XML-a i JSON-a.

4.7. JSON-LD

Povezani podaci način su stvaranja mreže podataka temeljenih na standardima koji pružaju mogućnost strojne interpretacije u različitim dokumentima i web stranicama. Taj način omogućava aplikacijama da započnu s jednim dijelom povezanih podataka i zatim prate ugrađene veze do ostalih dijelova povezanih podataka koji su rasprostranjeni na različitim web mjestima diljem weba (Kellogg, Champin i Longley, 2019). JSON-LD također je laganiji, ali koristi se za serijalizaciju povezanih podataka u JSON. Zbog njegovog dizajna, moguća je jednostavna interpretacija postojećeg JSON-a kao povezanih podataka uz minimalne promjene. „JSON-LD primarno je namijenjen da bude način korištenja povezanih podataka u programskim okruženjima temeljenim na webu, za izgradnju interoperabilnih web usluga i pohranjivanje povezanih podataka u strojeve za pohranu temeljene na JSON-u.“ (Kellogg, Champin i Longley, 2019, p. 7). Uzimajući u obzir da je JSON-LD stopostotno kompatibilan s JSON-om, brojni JSON raščlanjivači i knjižnice dostupni danas mogu biti ponovno upotrebljeni. Uz sve značajke koje JSON pruža, JSON-LD dodatno predstavlja (Kellogg, Champin i Longley, 2019):

- Mehanizam univerzalnog identifikatora za JSON objekte korištenjem IRI-a (*Internationalized Resource Identifier*)

- Način razjašnjenja ključeva koji se dijele između različitih JSON dokumenata tako da se preslikaju u IRI putem konteksta
- Mehanizam u kojem se vrijednost u JSON objektu može odnositi na resurs na drugoj stranici na webu
- Sposobnost označavanja nizova znakova svojim jezikom
- Način povezivanja tipova podataka s vrijednostima kao što su datum i vrijeme
- Mogućnost izražavanja jednog ili više usmjerenih grafova, poput društvene mreže, u jednom dokumentu

JSON-LD dizajniran je tako da se može koristiti izravno kao JSON, bez ikakvog znanja o RDF-u, a također se može koristiti i u kombinaciji s ostalim tehnologijama povezanih podataka, poput SPARQL-a (Kellogg, Champin i Longley, 2019). Sintaksa je osmišljena tako da ne ometa već postojeće implementirane sustave koji rade na JSON-u, već pruža glatku putanju nadogradnje s JSON-a na JSON-LD, a budući da oblik takvih podataka jako varira, JSON-LD ima mehanizme za preoblikovanje dokumenata u deterministički strukturu koja pojednostavljuje njihovu obradu (Kellogg, Champin i Longley, 2019). JSON-LD najjednostavnije je razumijeti na temelju jednostavnih usporednih primjera JSON i JSON-LD dokumenata. U nastavku se nalazi primjer jednostavnog JSON dokumenta kako su to prikazali Kellogg, Champin i Longley (2019):

```
{
    "name": "Manu Sporny",
    "homepage": "http://manu.sporny.org/",
    "image": "http://manu.sporny.org/images/manu.png"
}
```

Iz ljudske perspektive, jednostavno je za zaključiti da se u navedenom primjeru radi o osobi koja se zove „Manu Sporny“ te da pojam „homepage“ sadrži URL početne stranice te osobe. Međutim, strojevi nemaju jednako razumijevanje kao ljudi te je za njihovo razumijevanje ovakvih sadržaja potrebno koristiti vrstu jednoznačnih identifikatora. Povezani podaci koriste IRI-eve za jednoznačnu identifikaciju, pa se isti koriste i u JSON-LD formatu. Ako se uzme prethodni primjer JSON dokumenta, ali ovaj put ga se prikaže uz pomoć IRI-eva umjesto pojmova, dokument izgleda ovako (Kellogg, Champin i Longley, 2019):

```
{
    "http://schema.org/name": "Manu Sporny",
    "http://schema.org/url": {
        "@id": "http://manu.sporny.org/"
    },
    "http://schema.org/image": {
        "@id": "http://manu.sporny.org/images/manu.png"
    }
}
```

Svako svojstvo u primjeru iznad jednoznačno je identificirano uz pomoć IRI-a, a sve vrijednosti koje predstavljaju IRI-eve označene su uz pomoć ključne riječi „id“ Iako je ovo primjer regularnog JSON-LD dokumenta, dokument otežava rad za ljudske programere. JSON-LD rješava taj problem tako da uvodi pojam konteksta. Sam „context“ omogućava dvjema aplikacijama korištenje skraćenih pojmova za što učinkovitiju međusobnu komunikaciju, ali da se istovremeno ne gubi točnost (Kellogg, Champin i Longley, 2019). Ako se navedeno pojednostavi, kontekst u suštini služi za mapiranje pojmova u IRI-eve. Ako se na primjer iznad nadoda kontekst, dokument bi izgledao ovako (Kellogg, Champin i Longley, 2019):

```
{
    "@context": {
        "name": "http://schema.org/name",
        "image": {
            "@id": "http://schema.org/image",
            "@type": "@id"
        },
        "homepage": {
            "@id": "http://schema.org/url",
            "@type": "@id"
        }
    }
}
```

Iz navedenog primjera može se zaključiti da su „name“. „image“ te „homepage“ skraćenice za odgovarajuće dulje vrijednosti, dok se vrijednosti tipa niza znakova koje će biti povezane s „image“ i „homepage“ trebaju tumačiti kao identifikatori koji su IRI, a to je poznato na temelju “@type“:“@id“ dijela dokumenta. Konteksti se mogu ugraditi izravno u dokument, ali na njih se također može i referencirati što programerima omogućava ponovno korištenje međusobnih podataka bez prethodnog dogovora na koji način će njihovi podaci međusobno funkcionirati za svaku pojedinačnu stranicu (Kellogg, Champin i Longley, 2019).

5. Gettyjev tezaurus za umjetnost i arhitekturu

AAT (*The Art & Architecture Thesaurus*) najpoznatiji je tezaurus za područje umjetnosti i arhitekture. Sam tezaurus strukturirani je rječnik koji uključuje razne termine, opise i druge metapodatke za različite objekte, koncepte, umjetnike kao i mjesta važna za veliki broj različitih disciplina specijaliziranih za područje umjetnosti, arhitekture i ostale materijalne kulture (The Getty Research Institute, 2017). AAT ima kontinuirani rast upravo zbog doprinosa velikog broja korisnika.

5.1. Povijest AAT-a

Već 1970-ih godina razne umjetničke knjižnice i servisi za indeksiranje umjetničkih časopisa počeli su automatizirati svoje postupke katalogizacije i indeksiranja te je posljedično tome započeo rad na AAT-u. Od samog početka, AAT je dizajniran za različite skupine korisnika. Izvorna jezgra AAT-a prikupljena je iz autoritativnih popisa i pojmova koji se već koriste u literaturi o povijesti umjetnosti i arhitekture, a navedenu terminologiju odobrio je i dopunio znanstveni savjetodavni tim koji se sastoji od povjesničara umjetnosti i arhitekture, arhitekata, knjižničara, kustosa vizualnih izvora, arhivista, muzejskih djelatnika i stručnjaka za izradu tezaurusa (The Getty Research Institute, 2017). Postići suglasnost između navedenih različitih skupina ljudi bio je uspješan, ali isto tako i dugotrajan proces s iznimno zahtjevnim pregovorima, a sve s ciljem kreiranja jedinstvenog izvora koji bi odgovarao interesima svake skupine.

AAT se održava i konstruira prema temeljnim načelima koja su određena 1981. godine, a praktični dio morao je s vremenom biti ažuriran kako bi se prilagodio sve

većoj višejezičnosti i multikulturalnosti te kako bi povećao inkluzivnost (The Getty Research Institute, 2017). Povećanje inkluzivnosti bilo je od velike važnosti upravo zato što je AAT resurs koji je dio svog rasta postigao brojnim doprinosima od strane raznih zajednica korisnika. Navedeni doprinosi danas još uvijek AAT-u omogućavaju kontinuirani rast.

5.2. Opseg i struktura

"AAT uključuje generičke pojmove i povezane datume, odnose i druge informacije o konceptima koji se odnose ili su potrebni za katalogiziranje, otkrivanje i dohvaćanje informacija o umjetnosti, arhitekturi i drugoj vizualnoj kulturnoj baštini, uključujući srodne discipline koje se bave vizualnim djelima, kao što su arheologija i konzervacija, gdje se radi o djelima koja prikupljaju muzeji umjetnosti i repozitoriji za vizualnu kulturnu baštinu ili su to djela arhitekture" (The Getty Research Institute, 2017). Isto tako, uzimajući u obzir raznovrsne zbirke koje se mogu pronaći u muzejima umjetnosti, bitno je napomenuti da AAT također posjeduje terminologiju uz pomoć koje se opisuju predmeti i aktivnosti različitih priroda koje prema tradicionalnom zapadnom konceptu često nisu označene kao umjetnost (The Getty Research Institute, 2017).

Iako AAT uključuje mnogobrojne različite sadržaje, primarni doseg limitiran je na domenu vizualnih umjetnosti pa je tako terminologija iz drugih domena općenito isključena iz AAT-a. Međutim, iako su ostale domene isključene, njihova terminologija potrebna je do određene mjere, odnosno u slučajevima kada je ista neophodna za katalogiziranje ili otkrivanje određenih informacija o sadržajima iz domene vizualnih umjetnosti, što uključuje i novomedijsku umjetnosti, konceptualnu umjetnost i izvedbenu umjetnost (The Getty Research Institute, 2017). Iz AAT-a također su isključeni pojmovi koji su kombinacija riječi iz različitih hijerarhija (engl. *Hierarchy*), odnosno tzv. nevezani složeni pojmovi, primjerice "*gothic picture*". Taj pojam nevezani je složeni pojam koji je kombinacija stila i vrste djela pa je sukladno tome isključen. U suštini, isključeni su svi zapisi koji ne sadrže minimalne potrebne informacije da budu AAT zapis.

Nadalje, kao što je već spomenuto, AAT je strukturirani rječnik koji sadrži pojmove i druge informacije o konceptima, dok su u ciljanu publiku uključeni muzeji, knjižnice, zbirke vizualnih izvora, arhivi, konzervatorski projekti, projekti katalogizacije te bibliografski projekti (The Getty Research Institute, 2017). Upravo su koncepti fokus zapisa u AAT-u. Dakle, koncepti se odnose se na zapise u AAT-u koji predstavljaju koncepte, a koji uključuju većinu pojmova u AAT-u, odnosno vrste objekata i arhitekture, materijale, stilove i razdoblja, vrste ljudi, aktivnosti, fizičke atribute i povezane koncepte (The Getty Research Institute, 2017). Uz svaki koncept povezani su pojmovi, povezani koncepti, "roditelj" (engl. parent), odnosno pozicija u hijerarhiji, izvori podataka te bilješke, a pojmovi za bilo koji koncept mogu uključivati množinu, jedninu, prirodni redoslijed, obrnuti redoslijed, varijacije pisanja, znanstvene i uobičajene oblike, različite fonetske oblike i sinonime s različitim etimologijama (The Getty Research Institute, 2017). Važno je za napomenuti da među tim pojmovima samo je jedan označen kao preferirani termin ili deskriptor, ali uzimajući u obzir i upotrebu u više jezika, može ih potencijalno biti i više.

5.2.1. Fasete i hijerarhije

AAT je hijerarhijska baza podataka, ali može postojati i više širih konteksta pa je prema tome AAT polihijerarhijski. Međutim, osim takvih hijerarhijskih odnosa, AAT također sadrži i ekvivalentne te asocijativne odnose (The Getty Research Institute, 2017). Glavnu podpodjelu stukture AAT-a čine fasete (engl. facet). Fasete sadrže istovrsnu klasu koncepata čiji članovi dijele određene karakteristike prema kojima se razlikuju od članova ostalih klasa (The Getty Research Institute, 2017). Kao primjer može se navesti pojam "*renaissance*" koji predstavlja jedinstveni umjetnički stil te se prema tome nalazi u Stilovi i razdoblja faseti (engl. *Styles and Periods facet*). Kao još jedan primjer može se uzeti pojam "*artist*" koji predstavlja osobu koju identificiramo sa djelatnošću unutar domene umjetnosti te se isti nalazi u Agenti faseti (engl. *Agents facet*).

5.2.1.1. Fasete

Fasete su konceptualno organizirane u shemu koja se kreće od apstraktnih pojmova do konkretnih, fizičkih artefakata, a u nastavku slijedi cjelokupini popis faseta unutar AAT-a (The Getty Research Institute, 2017):

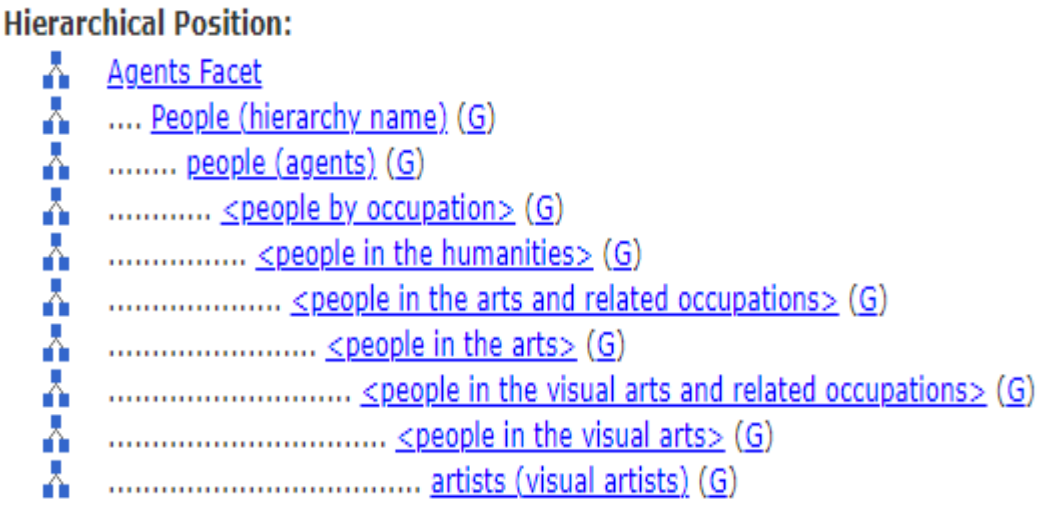
- Povezani pojmovi (engl. *Associated Concepts*) – ova faseta sadrži termine za apstraktne pojmove i pojave koji se odnose na proučavanje i izvođenje različitih područja ljudske misli i aktivnosti, uključujući arhitekturu i umjetnost u svim medijima, kao i srodne discipline. Ovdje su također obuhvaćena teorijska i kritička pitanja, ideologije, stavovi i društveni ili kulturni pokreti, u mjeri potrebnoj za katalogiziranje i otkrivanje informacija o vizualnim djelima
- Fizički atributi (engl. *Physical Attributes*) – ova faseta sadrži pojmove za uočljive ili mjerljive karakteristike materijala i artefakata kao i značajke materijala i artefakata koji se ne mogu odvojiti kao komponente, a potrebni su za katalogiziranje ili otkrivanje informacija o vizualnim djelima. Uključene su karakteristike kao što su veličina i oblik, kemijska svojstva materijala, kvalitete teksture i tvrdoće i značajke kao što su površinski ukrasi
- Stilovi i razdoblja – ova faseta sadrži nazive umjetničkih i arhitektonskih stilova, povijesnih razdoblja, umjetničkih pokreta, kultura i etničkih pripadnosti. Imena ljudi i mjesta su uključena ako označavaju različite stilove ili razdoblja
- Agenti – ova faseta sadrži generičke pojmove koji označavaju ljude, grupe ljudi i organizacije identificirane prema zanimanju, aktivnosti ili drugim karakteristikama kao što su fizičke ili mentalne karakteristike, društvena uloga ili stanje, a koji su povezani s umjetnošću, arhitekturom i drugim vizualnim djelima. Također su uključeni i pojmovi za druge žive organizme
- Aktivnosti (engl. *Activities*) – ova faseta sadrži pojmove za područja nastojanja, fizičke i mentalne radnje, diskretne pojave, sustavne sekvence radnji, metode korištene prema određenom cilju i procese koji se odvijaju u materijalima ili vizualnim djelima. Aktivnosti mogu varirati od grana učenja i profesionalnih područja do specifičnih životnih događaja, od mentalno izvršenih zadataka do procesa koji se izvode na ili s materijalima i predmetima, od pojedinačnih fizičkih radnji do složenih igara
- Materijali (engl. *Materials*) – ova faseta sadrži izraze za fizičke tvari, bilo da su prirodno ili sintetski dobivene. Oni se kreću od specifičnih materijala do vrsta materijala dizajniranih prema njihovoj funkciji, kao što su bojila, a zatim i od sirovih materijala do onih koji su oblikovani ili prerađeni u proizvode koji se koriste u izradi umjetničkih, arhitektonskih ili drugih vizualnih djela
- Objekti (engl. *Objects*) – ova faseta je najveća od svih AAT faseta. Sadrži izraze za diskretne opipljive ili vidljive stvari koje su nežive i proizvedene

ljudskim naporima, odnosno koji su ili proizvedeni ili im je dat oblik ljudskom aktivnošću. Oni se kreću, u fizičkom obliku, od izgrađenih radova do slika i pisanih dokumenata. Njihova namjena varira od utilitarne do estetske. Također su uključene značajke krajolika koje pružaju kontekst za izgrađeni okoliš

- Nazivi robnih marki (engl. *Brand Names*) – ova faseta dopušta potrebne dodatke od strane konzervatorske zajednice, posebno kada materijal, proces ili objekt nemaju generički naziv i nazivi su zaštićeni zaštitnim znakom. Navedeno je uključeno prema potrebi za katalogiziranje i otkrivanje informacija o vizualnim djelima

5.2.1.2. Hijerarhije

Istovrsne terminološke skupine, odnosno hijerarhije, raspoređene su unutar osam faseta AAT-a. Širi pojam daje neposrednu klasu ili rod pojmu te služi za dodatno pojašnjenje njegovog značenja, dok je uži pojam uvijek tip, vrsta, primjer ili manifestacija njegovog šireg konteksta (The Getty Research Institute, 2017). Za primjer se može uzeti Slika 1. iz koje je vidljivo da je pojam "*people in the visual arts*" širi kontekst za pojam "*artists (visual artists)*" jer su svi vizualni umjetnici ljudi u vizualnim umjetnostima.



Slika 3. Primjer fasete s hijerarhijom za pojam "*artists (visual artists)*". www.getty.edu. (n.d.). *Art & Architecture Thesaurus Full Record Display (Getty Research)*. Dostupno na: http://vocab.getty.edu/page/aat/300025103 (Pristupljeno 15.08.2022.)

Uzimajući u obzir da su hijerarhije raspoređene unutar faseta, u nastavku slijedi popis faseta s nazivima svojh odgovarajućih hijerarhija, kao što je to naveo The Getty Research Institute (2017), uz napomenu da je faseta Objekti zbog svoje opsežnosti specifična te određene hijerarhije sadrže i podhijerarhije:

- Povezani pojmovi faseta
    - Hijerarhija: Povezani pojmovi
- Fizički atributi faseta
    - Hijerarhije: Atributi i svojstva (engl. *Attributes and Properties*), Stanja i učinci (engl. *Conditions and Effects*), Elementi dizajna (engl. *Design Elements*), Boja (engl. *Color*)
- Stilovi i razdoblja faseta
    - Hijerarhija: Stilovi i razdoblja
- Agenti faseta
    - Hijerarhije: Ljudi (engl. *People*), Organizacije (engl. *Organizations*), Živi organizmi (engl. *Living Organisms*)
- Aktivnosti faseta
    - Hijerarhije: Discipline (engl. *Disciplines*), Funkcije (engl. *Functions*), Događaji (engl. *Events*), Fizičke i mentalne aktivnosti (engl. *Physical and Mental Activities*), Procesi i tehnike (engl. *Processes and Techniques*)
- Materijali faseta
    - Hijerarhija: Materijali
- Objekti faseta
    - Hijerarhije: Grupacije objekata i sustavi (engl. *Object Groupings and Systems*), Žanrovi objekata (engl. *Object Genres*), Komponente (engl. *Components*)
    - Izgrađeni okoliš (engl. *Built Environment*) hijerarhija
        - Hijerarhije: Naselja i krajolici (engl. *Settlements and Landscapes*), Izgrađeni kompleksi i četvrti (engl. *Built Complexes and Districts*), Pojedinačna izgrađena djela (engl. *Single Built Works*), Otvoreni prostori i elementi mjesta (engl. *Open Spaces and Site Elements*)

- Namještaj i oprema (engl. *Furnishings and Equipment*) hijerarhija
    - Hijerarhije: Namještaj, Kostimi (engl. *Costume*), Alati i oprema (engl. *Tools and Equipment*), Oružje i streljivo (engl. *Weapons and Ammunition*), Mjerni uređaji (engl. *Measuring Devices*), Spremnici (engl. Containers), Zvučni uređaji (engl. *Sound Devices*), Rekreacijski artefakti (engl. *Recreational Artifacts*), Prijevozna sredstva (engl. *Transportation Vehicles*)
- Vizualna i verbalna komunikacija (engl. *Visual and Verbal Communication*) hijerarhija
    - Hijerarhije: Vizualna djela (engl. *Visual Works*), Mediji razmjene (engl. *Exchange Media*), Obrasci za informacije (engl. *Information Forms*)

- Nazivi robnih marki faseta
    - Hijerarhija: Nazivi robnih marki

5.3. AAT i LOD/JSON-LD

The Getty Vocabularies struktura oduvijek je bila osmišljena na način koji omogućava međusobno povezivanje. A uz pomoć LOD projekta takvo međusobono povezivanje se i ostvarilo. *Getty Vocabularies* cijenjeni su kao LOD iz više razloga (Harpring et al, 2018):

- Kvaliteta je pouzdana jer je sastavljena od strane stručnjaka, a izvori su citirani
- Podaci su povezivi
- Od početka su planirani da budu povezani i da pomognu u odgovaranju na složena pitanja povijesti umjetnosti, kao primjerice povratni podaci za uljane slike koje se sada nalaze u Sjevernoj Americi i imaju nekršćansku ikonografiju, a originalno su ih stvorili nizozemski ili francuski slikari koji su bili aktivni u Toskani ili Umbriji u Italiji između 1410. i 1630. godine
- Sadrže tezauralne i druge bogate veze
- Jedinstveni ID-evi za zapise, termine i ostale podatke
- Uključuju i ID-eve vanjskih izvora kako bi se omogućilo povezivanje

Svaka stvar, kao na primjer neki određeni muzejski predmet ili osoba, mora biti predstavljena jednoznačnim identifikatorom, a u AAT-u se koristi URI (*Uniform Resource* Identifier). Također, AAT koristi više LOD formata, a među njima nalaze se JSON i JSON-LD formati. U nastavku se nalazi jedan primjer AAT-a u JSON-LD formatu (The Getty Research Institute, 2022):


```
{
 "@context": "https://linked.art/ns/v1/linked-art.json",
 "id": "http://vocab.getty.edu/aat/300198841",
 "type": "Type",
 "subject_of": [
  {
   "id": "http://vocab.getty.edu/aat/scopeNote/137203",
   "type": "LinguisticObject",
   "content": "Verweist auf ein Gefäß aus dem antiken Griechenland, Osteuropa oder dem Mittleren Osten, das typischerweise eine geschlossene Form mit zwei Öffnungen hat, eine auf der Oberseite zum Einfüllen, eine auf der Unterseite, damit die Flüssigkeit herauslaufen kann. Es hat oft die Form eines Horns oder Tierkopfes und wurde typischerweise als Trinkgefäß oder zum Ausgießen von Wein in ein anderes Gefäß benutzt. Getrunken wurde, indem das Rhyton über den Kopf des Trinkenden gehalten und die Flüssigkeit mit dem Mund aufgefangen wurde.",
   "classified_as": [
    {
     "id": "http://vocab.getty.edu/aat/300435416",
     "type": "Type",
     "_label": "descriptive note"
    }
   ],
   "referred_to_by": [
    {
     "id": "http://vocab.getty.edu/aat/300198841#description-300198841-contributor-10000232",
     "type": "Name",
     "content": "Staatliche Museen zu Berlin Preussischer Kulturbesitz (Berlin, Germany)",
     "identifies": "http://vocab.getty.edu/aat/contrib/10000232",
     "classified_as": [
      {
```

```
            "id": "http://vocab.getty.edu/aat/300403974",
            "type": "Type",
            "_label": "contributors"
          }
        ]
      },
          {
            ...
          }
}
```


Iz navedenog primjera može se uočiti uporaba ranije spomenutog konteksta koji omogućava korištenje skraćenih pojmova za što lakšu daljnju komunikaciju između aplikacija, ali još uvijek održava točnost s obzirom na to da se koriste URI-evi. Isto tako, primjer se može vrlo jednostavno razumijeti od strane ljudi, pa se tako može zaključiti da se ovaj primjer odnosi na posudu iz antičke Grčke, istočne Europe ili Bliskog istoka koja se obično koristila kao posuda za piće ili za pretakanje vina u drugu posudu te se trenutno nalazi u muzeju u Berlinu u Njemačkoj.

6. Zaključak

Želja za što bržim pristupom željenim relevantnim podacima postavila je metapodatke u fokus mnogih institucija i pojedinaca. Svega nekoliko metapodataka o nekom sadržaju mogu značajno povećati sveopću dostupnost tog sadržaja prema korisnicima. Iako metapodaci imaju velik značaj, kod povezanih otvorenih podataka mogu se ponekad smatrati dodatnom aktivnošću s velikom potrošnjom resursa. Sami metapodaci zahtijevaju određenu količinu vremena kako bi se maksimizirala njihova korist, a zahtijevaju i relativno velika ulaganja u pogledu samog održavanja i kontinuiranog unaprjeđenja. Metapodaci imaju svoje prednosti i nedostatke te je potrebno pažljivo održavanje ravnoteže između tih prednosti i mana kako bi se metapodaci maksimalno iskoristili uz što manje troškove. Gettyjev tezaurus za umjetnost i arhitekturu odličan je primjer ekstremno kvalitetne strukture metapodaka i otvorenih povezanih podataka koji uključuje razne termine, opise i druge metapodatke za različite objekte, koncepte, umjetnike i ostale aspekte iz područja umjetnosti, arhitekture i ostale materijalne kulture. Kako bi se ti metapodaci i povezani otvoreni podaci uspješno mogli dijeliti putem različitih aplikacija, različitih domena ili nekim drugim spremištima podataka, potrebni su nam procesi serijalizacije i deserijalizacije, a osobito XML i JSON oblici. U razvoju web aplikacija, XML i JSON serijalizacije prevladavaju nad ostalim oblicima jer mogu biti međusobno zamjenjivi, što značajno poboljšava komunikaciju između aplikacija. XML i JSON također imaju svoje prednosti i nedostatke te nisu uvijek korisni u svakoj situaciji. XML je odličan izbor kada se prenose brojne različite vrste podataka, dok je JSON idealan za jednostavnije prijenose podataka. Danas se često razvijaju dinamičke aplikacije za koje je JSON puno prikladniji. Upravo zbog toga, JSON je trenutno u puno većoj upotrebi od XML-a te se često postavlja pitanje hoće li JSON u potpunosti zamijeniti XML. Iako je XML trenutno u manjoj upotrebi od JSON-a, JSON ne može u potpunosti zamijeniti XML na webu. XML će zbog svojih bogatih karakteristika u aspektima sigurnosti i validacije dokumenata uspješno zadržati svoje mjesto na webu. Međutim, postoji mogućnost da će se u budućnosti razvijati nove vrste tehnologija u kojima će XML ponovno doći do izražaja, ali uzimajući u obzir trenutno stanje JSON popularnosti, nije izgledno da će se to dogoditi u bližoj budućnosti.

7. Literatura

1. Asad, A. i Ali, H. (2017) *The C# Programmer's Study Guide (MCSD) Exam: 70-483*. 1st edn. Berkeley: Apress.
2. Bassett, L. (2015). *Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON*. 1st edn. Sebastopol: 'O'Reilly Media, Inc.
3. Berners-Lee, T. J. (2006) *Linked Data – Design Issues*. Dostupno na: https://www.w3.org/DesignIssues/LinkedData.html (Pristupljeno 03.08.2022.)
4. Braunschweig, K., Eberius, J., Thiele, M., i Lehner, W. (2012) *The State of Open Data, Limits of Current Open Data Platforms.* Dostupno na: https://wwwdb.inf.tu-dresden.de/opendatasurvey/www2012_short.pdf (Pristupljeno 03.08.2022.)
5. Bray, T. (2014). The JavaScript Object Notation (JSON) Data Interchange Format. Dostupno na: https://www.rfc-editor.org/rfc/rfc7159 (Pristupljeno 20.08.2022.)
6. Coyle K. (2012) Chapter 2: Semantic Web and Linked Data, *Library Technology Reports*, 48(4), pp. 10-14. Dostupno na: https://www.journals.ala.org/index.php/ltr/article/view/4669/5538 (Pristupljeno 04.08.2022.)
7. Haq, Z., Faraz Khan, G. i Hussain, T. (2013). *A Comprehensive analysis of XML and JSON web technologies.* Dostupno na: http://inase.org/library/2015/vienna/bypaper/CSSCC/CSSCC-14.pdf (Pristupljeno 17.08.2022.)
8. Harpring, P., Cobb, J., Garcia, G., Zeng, M. i Alexiev V. (2018) *Introduction to LOD*. Dostupno na: https://www.getty.edu/research/tools/vocabularies/Linked_Data_Getty_Vocabularies.pdf (Pristupljeno 25.08.2022.)
9. Joshi, B. (2017). *Beginning XML with C# 7*. 2nd edn. Berkeley: Apress.
10. Kellogg, G., Champin, P.-A. i Longley, D. (2019). *JSON-LD 1.1 – A JSON-based Serialization for Linked Data*. Dostupno na: https://hal.archives-ouvertes.fr/hal-02141614/ (Pristupljeno 25.08.2022.)
11. Lubas, R., Jackson, A., i Schneider, I. (2013) *The Metadata Manual: A Practical Workbook*. Witney: Chandos Publishing.
12. National Information Standards Organisation (NISO). (2004) Understanding Metadata. Dostupno na: http://arizona.openrepository.com/arizona/handle/10150/105486 (Pristupljeno 03.08.2022.)
13. Pérez, J., Arenas, M., i Gutierrez, C. (2009) Semantics and complexity of SPARQL, *ACM Transactions on Database Systems (TODS),* 34(3), pp. 1-45. Dostupno na: https://dl.acm.org/doi/abs/10.1145/1567274.1567278 (Pristupljeno 04.08.2022.)
14. Pezoa, F., Reutter, J.L., Suarez, F., Ugarte, M. i Vrgoč, D. (2016). Foundations of JSON Schema. *Proceedings of the 25th International*

Conference on World Wide Web. Dostupno na: https://dl.acm.org/doi/abs/10.1145/2872427.2883029 (Pristupljeno 20.08.2022.)
15. The Getty Research Institute. (2017) *Art & Architecture Thesaurus Online*. Dostupno na: https://www.getty.edu/research/tools/vocabularies/aat/about.html (Pristupljeno 10.08.2022.)
16. The Getty Research Institute. (2022) *Getty Vocabularies LOD Semantic Resolution*. Dostupno na: http://vocab.getty.edu/LinkedArtSemanticResolution Pristupljeno (20.08.2022.)
17. Wang, G. (2011). *Improving Data Transmission in Web Applications via the Translation between XML and JSON*. IEEE Xplore. Dostupno na: https://ieeexplore.ieee.org/abstract/document/5931189 (Pristupljeno 17.08.2022.)
18. Yu, L. (2011) *A Developer's Guide to the Semantic Web*. Springer Science & Business Media.

8. Popis slika

Serijalizacija metapodataka na primjeru Gettyjevog Tezaurusa za umjetnost i arhitekturu

Sažetak

Metapodaci su podaci o podacima koji se, u najširem smislu, koriste za opisivanje resursa informacija, što bi značilo da se koriste na internetu, kao i u knjižnicima, arhivima i drugim ustanovama. Povezani otvoreni podaci (LOD) povezani su podaci za višekratnu besplatnu upotrebu koji su objavljeni i dostupni svima. Proces serijalizacije je proces pretvaranja objekta u bajtove ili tekst kako bi se pohranio u bilo koju vrstu pohrane ili razmijenio objekt preko mreže, a služi za što jednostavniju razmjenu podataka između različih repozitorija podataka. XML i JSON formati su jedni od najpoznatijih i najčešće korištenih formata serijalizacije. Gettyjev tezaurus za umjetnost i arhitekturu (AAT) najpoznatiji je tezaurus za područje umjetnosti i arhitekture. Sam tezaurus strukturirani je rječnik koji uključuje razne termine, opise i druge metapodatke, a ujedno je i jako kvalitetan primjer strukture povezanih otvorenih podataka koja se može prikazati uz pomoć JSON-LD formata.

Ključne riječi: metapodaci, povezani otvoreni podaci (LOD), serijalizacija, XML, JSON, AAT, JSON-LD

Serialization of metadata using the example of The Getty Thesaurus for Art and Architecture

Summary

Metadata is information about data that, in the broadest sense, is used to describe information resources, which would mean that they are used on the Internet, as well as in libraries, archives and other institutions. Linked Open Data (LOD) is linked data for repeated free use that is published and available to everyone. The process of serialization is the process of converting an object into bytes or text in order to store it in any type of storage or to exchange the object over the network, and it serves to make data exchange between different data repositories as simple as possible. XML and JSON formats are some of the most well-known and widely used serialization formats. The Getty Thesaurus for Art and Architecture (AAT) is the most famous thesaurus for the field of art and architecture. The thesaurus itself is a structured dictionary that includes various terms, descriptions and other metadata, and is also a very high-quality example of the structure of linked open data that can be displayed using the JSON-LD format.

Key words: metadata, linked open data (LOD), serialization, XML, JSON, AAT, JSON-LD