Projektim dhe zhvillim në RESTful API i një shërbimi Web që ofron të dhëna në kod të hapur

Sidita Duli, Departamenti i Matematikës, Universiteti "Luigj Gurakuqi", Shkodër
siditaduli@yahoo.com

PËRMBLEDHJE

Në ditët e sotme, shërbimet në Web marrin rëndësi të veçantë për gamën e gjerë të aplikacioneve të implementuara si dhe për sasinë e të dhënave që kërkojnë përpunim. Të dhënat publike në kod të hapur iu ofrohen për përpunim aplikacioneve të ndryshme në platforma të ndryshme Web ose mobile. Ky artikull përfshin një pasqyrë të përgjithshme të RESTful API të implementuar në JSON, duke përfshirë përfitimet dhe sfidat e tij. Nëpërmjet një implementimi konkret do të tregohet se si të dhënat e ruajtura në format JSON, ne kod të hapur, ndihmojnë në rritjen e performancës së shërbimeve të bizneseve që do i përdorin këto të dhëna.

Në këtë artikull janë përshkruar në detaje metodat dhe teknologjitë e Java API për RESTful Web Service si dhe librarinë Jersey, të gjitha këto nën protokollin HTTP. Mbi këto platforma është implementuar aplikacioni Web i këtij studimi, i cili konsiston në një RESTful Web Service që transformon të dhënat e shërbimit meteorologjik pranë Universitetit të Shkodrës, nga format Excel në format JSON dhe i vë në dispozicion me kod të hapur. Ky punim thekson rëndësinë dhe rolin e aplikacioneve dhe shërbimeve Web me të dhëna në kod të hapur.

Fjalë kyçe : RESTful API, Web Service, JSON, Java

Design and Build a RESTful API Web Service which offers open-source data

ABSTRACT

Nowadays, Web services have become more popular because of the wide range of applications which use them and the amount of data that requires analysis. The open-source data are offered to various applications on different Web or mobile platforms. This article introduces a general overview of the RESTful API implemented on JSON, and sumerizes the challenges. It describes how a Web Service enables businesses to exchange data on different Web platforms. Through the implementation, this research highlights the role of the open-source data in JSON improves the performance of online business that will use these data.

This article describes in detail the Java API technology for RESTful Web Service as well as the Jersey library, following the HTTP protocol. The web application part this study consists of a RESTful Web Service that transforms the meteorological data of the Weather station of University of Shkodra from Excel format to JSON format and makes them available in open source. This paper highlights the role of open source applications.

Key words: RESTful API, Web Service, JSON, Java

HYRJE

Termi Application Programming Interface (API) përdoret për të përshkruar cilësitë e një librarie dhe veprimet që ajo kryen. Kjo librari mund të ketë një “API Documentation” i cili është një dokumentacion mbi funksionet e librarisë, argumentet që kërkohen, si realizohen thirrjet, etj. Në ditët e sotme, termit API shpesh i referohemi për një HTTP API, e cila lejon që aplikacione të ndryshme të shkëmbejnë të dhëna nëpërmjet Internet-i (Louvel et al., 2012).

Teknologjia API nuk është e re. Ajo ka shërbyer për vite me radhë në komunikimin e aplikacioneve të platformave të ndryshme, por roli i saj ka ndryshuar këto vitet e fundit. Kompani të ndryshme kanë vënë re se API mund të përdoret si një ndërfaqe biznesi, duke i ndërlidhë konsumatorët nëpërmjet kanaleve të ndryshme dhe të pajisjeve të ndryshme.

Kur ju krijoni një API, ju lejoni që persona të tjerë jashtë organizatës tuaj të marrin shërbimin ose produktin, si dhe të tërheqin konsumatorët dhe të zgjerojnë biznesin e tyre.

Një API e brendshme rrit produktivitetin e kompanisë duke rritur konsistencën e aplikacioneve të reja. Një API publike mund t’i shtojë vlera biznesit tuaj, duke bërë të mundur që zhvillues të tretë të përdorin shërbimet tuaja ose të sjellin klientët e tyre tek ju.

Një Web Service është një set protokollesh dhe standardesh të përdorura për shkëmbim të dhënash mes aplikacionesh ose sistemesh. Këto aplikacione mund të jenë duke u ekzekutuar në platforma të ndryshme, por Web Service bën që ato të përdorin rrjetin për shkëmbim informacioni, njësoj si të ishin në të njëjtin kompjuter. Ky bashkëveprim mes platformash, për shembull Java dhe Python, Windows dhe Linux, bëhet i mundur nga përdorimi i standardeve në kod të hapur.

Web services që bazohen në arkitekturën REST njihen si RESTful Web Services. Këto shërbime implementojnë konceptet e arkitekturës REST, duke përdorë metoda të protokollit HTTP. Një RESTful Web Service përfaqësohet nga një URI (Uniform Resource Identifier), dhe është shërbimi Web që ofron të dhëna JSON mbi protokollin HTTP.

Arkitektura REST konsideron shumë të rëndësishëm konceptin e resursit. Çdo përmbajtje është një resurs (Allamaraju, 2010). Ai mund të jetë një skedar tekst, një faqe HTML, video, etj. Ana e serverit në arkitekturën REST ofron akses në këto resurse, ndërsa ana e klientit REST lexon dhe/ose modifikon këto resurse. Çdo resurs identifikohet nga URI e tij. REST përdor disa mënyra prezantimi të resursit, ndër të tjera formatin XML dhe JSON. Në këtë punim, të dhënat paraqiten në formatin JSON.

Një resurs në REST ngjason me një objekt në gjuhët e programimit të orientuar nga objektet, ose me një entitet në një bazë të dhënash relacionale. Pasi resursi identifikohet me URI, dhe pasi është formatuar sipas një standardi, serveri REST ia dërgon klientit, i cili gjithashtu kupton këtë format të dhëne.

PLATFORMAT DHE METODA

Më poshtë përshkruhen disa nga platformat dhe libraritë e përdorura në këtë punim, duke cilësuar edhe avantazhet si dhe arsyet e përzgjedhjes së tyre:

Java API për Restful Web Services (JAX-RS) është një API në gjuhën e programimit Java i cili ofron suport në krijimin e Web Service REST (Richardson et al., 2013). JAX-RS është një koleksion me ndërfaqe dhe kode në Java, të cilat e thjeshtojnë procesin e zhvillimit të aplikacioneve Server-side REST.

Më poshtë jepen disa cilësi të aplikacioneve REST:

- *Performanca :* Mënyra e komunikimit të REST është eficiente dhe e thjeshtë, duke bërë që performanca e sistemit të rritet ndjeshëm.

- *Modifikimi i komponentëve:* Vetë natyra e shpërndarë e sistemit, bën të mundur që komponentë të REST të modifikohen në mënyrë të pavarur nga njëri-tjetri, me një kosto minimale.
- *Portabiliteti :* RESTful është një teknologji e cila mund të implementohet dhe të përdoret nga shumica e teknologjive aktuale.
- *Sistem i qëndrueshëm:* Fakti që shërbimi nuk ruan gjendjen, i jep mundësi për riparim të shpejtë pas një dështimi të sistemit.

Zhvillimi i RESTful Web Service që mbështet paraqitjen e të dhënave në një varietet formatesh multimediale, njëkohësisht pa detaje të komunikimit klient-server, bëhet i mundur nga një set mjetesh. Për të thjeshtuar fazën e zhvillimin të RESTful Web Service është projektuar libraria JAX-RS API, e mbështetur në Java.

Platforma Jersey për RESTful Web Service është me kod të hapur dhe ofron mundësinë e implementimit të RESTful Web Service në Java. Gjithashtu mbështet JAX-RS API si dhe mjete tjera të nevojshme për të thjeshtuar programimin e shërbimit RESTful (Newman 2015). Shkurtimisht, në fokus të një projekti në Jersey janë :

- Përfshirja e JAX-RS API për sigurimin e një produkti cilësor.
- Përfshirja e GlassFish si mjet implementues.
- Zhvillimi i një komuniteti përdoruesish të Jersey.
- Thjeshtimi i implementimit të RESTful Web Service, duke përdorë Java dhe Java Virtual Machine.

Formati JSON përdoret për serializimin dhe transmetimin në rrjet të të dhënave të strukturuara. Ai përdoret kryesisht për të transmetuar të dhëna mes një serveri dhe një aplikacioni Web. Web Service dhe API përdorin formatin JSON për të mundësuar të dhëna publike (Patni 2017).

Në këtë punim është përdorë ky format sepse është i thjeshtë në implementim, i pavarur nga gjuha e programimit të Web Service si dhe nuk kërkon shumë memorie shtesë për ruajtjen e të dhënave publike.

Protokolli HTTP lejon pjesëmarrjen e protokolleve të tjera në komunikimin përmes aplikacionit klient dhe serverit (Parastatidis et al., 2010). Katër protokollet shtesë mbi HTPP janë : GET, PUT, POST dhe DELETE.

GET , PUT dhe DELETE duhet të implementohen si protokolle të pandryshueshëm, në sensin që sado herë të përsëritet një kërkesë nga klienti, rezultati duhet të jetë i njëjtë. Përkundrazi, protokolli POST, nuk shfaq këtë cilësi.

Protokolli GET paraqitet në formën e një kërkese, e cila nuk përmban modifikim të gjendjes së sistemit, në asnjë formë të tij. Kjo nuk nënkupton që serveri nuk ndërmerr asnjë ndryshim, por se nuk është klienti ai që e kërkon ndryshimin. Realizimi i një kërkese GET përfshin kthimin e një rezultati, në këtë rast në format JSON.

Protokolli POST shfaqet në formën e një kërkese për krijimin e një entiteti të ri. Përmbajtja e këtij entiteti është pjesë e vetë kërkesës.

Protokolli PUT është i ngjashëm me POST, me ndryshimin se PUT krijon një entitet të ri në qoftë se nuk ekziston entiteti i përfshirë në kërkesë.

Protokolli DELETE shfaqet në formën e një kërkese për të fshirë një entitet të caktuar.

Kërkesa për Serverin
Lloji i veprimit që do kryhet specifikohet me anë të metodave HTTP : GET, POST, PUT dhe DELETE. Identifikuesi URI përcakton entitetin mbi të cilin kryhet veprimi. Në rastin e metodës GET, identifikuesi URI jep informacion shtesë në lidhje me tipin e kërkesës që serveri të filtrojë të dhënat. Këto parametra përfshihen në URI. Metodat e tjera (POST, PUT dhe DELETE) përmbajnë gjithë informacionin brenda trupit të mesazhit të formatuar me JSON.

Përgjigja nga Serveri
Përgjigja nga Serveri do jetë në formatin JSON dhe do përmbajë meta-data. Këto përgjigje do jenë të qëndrueshme, pra që t'i lejojë konsumatorit të API të dijë se çfarë të presë si përgjigje. Gjithashtu, i lejon programuesit të shkruajë më pak kod sepse disa përpunime të dhënash mund të kryhen në mënyrë të përgjithshme për çdo tip kërkese.

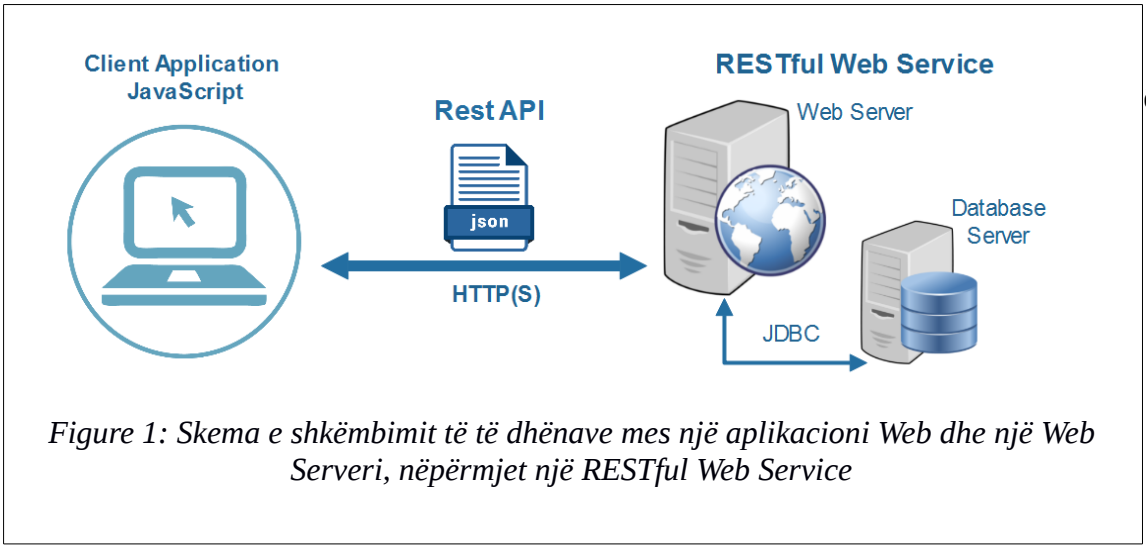

Figure 1: Skema e shkëmbimit të të dhënave mes një aplikacioni Web dhe një Web Serveri, nëpërmjet një RESTful Web Service

N ë figurën 1 paraqitet skema e komunikimit mes një aplikacioni Web, në rolin klient, dhe një Web Serveri. Aplikacioni klient mund të jetë një aplikacion në JavaScript, ose në ndonjë teknologji tjetër front-end. Web Server i ofron të dhënat në kod të hapur. Në këtë komunikim Klient-Server, libraria REST API luan një rol ndërmjetës. Kjo librari në Java ofron ndërmjetësimin nga aplikacioni front-end për lexim të dhënash në server. Kërkesa e klientit ka formatin e një thirrjeje në HTTP(S), ndërsa përgjigja e Web Server-it mund të jetë në format XML ose JSON. RESTful Web Service Ky model komunikimi ndan ndërfaqes së përdoruesit, ku përfshihen kërkesat dhe specifikimet, me logjikën e një modeli universal të dhënash.

IMPLEMENTIMI DHE DISKUTIMI
Pjesë integrale e këtij punimi është implementimi i një rasti konkret shërbimi Web, duke ofruar të dhëna në formatin JSON. I konceptuar si një shërbim në favor të shumë bizneseve lokale apo rajonale, të interesuar mbi të dhënat meteorologjike në Shkodër, është parë me interes ofrimi i këtyre të dhënave në kod të hapur, në format JSON. Të dhënat janë ofruar nga Dega Teknike Mësimore e Universitetit "Luigj Gurakuqi", Shkodër. Këto të dhëna aktualisht janë në format skedar në Excel, i cili nuk ofron fleksibilitet përpunimi për platformat Web.

Aplikacioni Web i projektuar dhe i zhvilluar në këtë punim përfshin një RESTful Web Service në Java, i cili lexon të dhënat nga skedar Excel dhe i kthen përgjigje aplikacionit klient në format JSON.
Me rëndësi janë diskutimet mbi skedarët :

- skedarin e konfigurimeve pom.xml
- skedarin enkapsulues të të dhënave TeDhenaMeteoUnishk.java
- skedarin ndihmës FormatiNeExcel.java
- skedarin e shërbimit web MeteoWebService.java

Skedari pom.xml paraqitet si më poshtë :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>RestWebServiceUnishk</groupId>
    <artifactId>RestWebServiceUnishk</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <build>
        <sourceDirectory>src</sourceDirectory>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.7.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.0.0</version>
                <configuration>
                    <warSourceDirectory>WebContent</warSourceDirectory>
                </configuration>
            </plugin>
        </plugins>
  </build>
</project>
```

Këtij skedari pom.xml i shtohen varesitë :

- asm-3.3.1.jar
- jersey-bundle-1.19.4.jar
- jersey-json-1.9.1.jar
- jersey-server-1.8.jar
- json-20140107.jar

- poi-ooxml-3.17.jar

Klasa TeDhenaMeteoUnishk enkapsulon të dhëna të formatuara. Ajo ofron shërbimet e leximit (get) për secilën të dhënë, si dhe të modifikimit (set) për të dhënat.

```
public class TeDhenaMeteoUnishk {
    private String data;
    private String ora;
    private int temperatura;
    private int sasiReshje;
    private int shpejtesiEre;
    private String drejtimEre;

    public TeDhenaMeteoUnishk() {
    }
    public TeDhenaMeteoUnishk(String dt, String o, int temp, int sReshje, int sh, String dr)
    {
        this.data=dt;
        this.ora=o;
        this.temperatura = temp;
        this.sasiReshje = sReshje;
        this.shpejtesiEre = sh;
        this.drejtimEre = dr;
    }

    public String getData() {
        return data;
    }
    public void setData(String dt) {
        this.data = dt;
    }
    public String getOra() {
        return ora;
    }
    public void setOra(String o) {
        this.ora = o;
    }
    public int getTemperatura() {
        return temperatura;
    }
    public void setTemperatura(int temp) {
        this.temperatura = temp;
    }
    public int getSasiReshje() {
        return sasiReshje;
    }
    public void setSasiReshje(int s) {
        this.sasiReshje = s;
```

```
    }
    public int getShpejtesiEre() {
        return shpejtesiEre;
    }
    public void setShpejtesiEre(int sh) {
        this.shpejtesiEre = sh;
    }
    public String getDrejtimEre() {
        return drejtimEre;
    }
    public void setDrejtimEre(String dr) {
        this.drejtimEre = dr;
    }
    @Override
    public String toString() {
      return new StringBuffer("Temperatura ").append(this.temperatura)
        .append(" Sasi Reshjesh ").append(this.sasiReshje)
        .append(" Shpejtesi Ere ").append(this.shpejtesiEre).append(" Drejtim Ere ")
        .append(this.drejtimEre).toString();
    }
}
```

Klasa ndihmëse FormatiNeExcel, është implementim në Java i një konvertuesi nga një skedar në Excel në një skedar JSON.

```
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;

public class FormatiNeExcel {
    static final String FILE_NAME = "C:\\Users\\Sidita\\Documents\\MeteoUnishk.xlsx";
    static ArrayList<TeDhenaMeteoUnishk> tabelaEPlote=new
ArrayList<TeDhenaMeteoUnishk>();

    public static ArrayList<TeDhenaMeteoUnishk> lexoTeDhenaExcel( ){
      TeDhenaMeteoUnishk njeRecord=new TeDhenaMeteoUnishk();

      try {
        FileInputStream skedariExcel = new FileInputStream(new File(FILE_NAME));
        Workbook workbook = new XSSFWorkbook(skedariExcel);
        Sheet datatypeSheet = workbook.getSheetAt(0);
        Iterator<Row> iterator= datatypeSheet.iterator();
        Cell qelizaExcel;
```

```
            Iterator<Cell> qelizaAktuale;
            Row rreshtiExcel;

            while (iterator.hasNext()) {
                rreshtiExcel= iterator.next();
                qelizaAktuale = rreshtiExcel.iterator();
                qelizaExcel = qelizaAktuale.next();
                njeRecord.setData(qelizaExcel.getStringCellValue());
                qelizaExcel = qelizaAktuale.next();
                njeRecord.setOra(qelizaExcel.getStringCellValue());
                qelizaExcel = qelizaAktuale.next();
                njeRecord.setTemperatura((int)qelizaExcel.getNumericCellValue());
                qelizaExcel = qelizaAktuale.next();
                njeRecord.setSasiReshje((int)qelizaExcel.getNumericCellValue());
                qelizaExcel =qelizaAktuale.next();
                njeRecord.setShpejtesiEre((int)qelizaExcel.getNumericCellValue());
                qelizaExcel = qelizaAktuale.next();
                njeRecord.setDrejtimEre(qelizaExcel.getStringCellValue());

                tabelaEPlote.add(new TeDhenaMeteoUnishk
( njeRecord.getData(),njeRecord.getOra(),njeRecord.getTemperatura(),njeRecord.getSasiResh
je(),njeRecord.getShpejtesiEre(),njeRecord.getDrejtimEre()) );
            }

        } catch (FileNotFoundException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
        return tabelaEPlote;
      }
}
```

E gjithë pjesa më specifike për Web Service ndodhet e implementuar në klasën MeteoWebService në Java. Të dhënat e përftuara në JSON i ofrohen si shërbim WEB, nën protokollin GET. Fillimisht ky skedar është testuar në server lokal dhe shumë shpejt mund të kalojë si shërbim në internet.

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
@Path("/shkarkoJson")
```

```
public class MeteoWebService {
    static TeDhenaMeteoUnishk mtData;
    @GET
    @Produces("application/json")
    public Response shkarkoTeDhenatJSON() throws JSONException {
        ArrayList<TeDhenaMeteoUnishk> teDhenat;
        teDhenat=FormatiNeExcel.lexoTeDhenaExcel();
        TeDhenaMeteoUnishk rresht;
        JSONObject jsObj;
        String result = "@Te Dhena JSON(\"application/json\") \n\n";
        for(int index=0; index<teDhenat.size(); index++){
            rresht = teDhenat.get(index);
            jsObj = new JSONObject();
            jsObj.put("Data", rresht.getData());
            jsObj.put("Ora", rresht.getOra());
            jsObj.put("Temperatura", rresht.getTemperatura());
            jsObj.put("Sasi Reshje", rresht.getSasiReshje());
            jsObj.put("Shpejtesi Ere", rresht.getShpejtesiEre());
            jsObj.put("Drejtim Ere", rresht.getDrejtimEre());
            result=result + jsObj +"\n";
        }
        jsObj=null;
        return Response.status(200).entity(result).build();
    }
}
```

Një shembull i një rezultati që aplikacioni klient lexon është kodi i mëposhtëm në JSON.

```
@Te Dhena JSON("application/json")
{"Sasi Reshje":0,"Drejtim Ere":"W","Temperatura":17,"Shpejtesi Ere":1,"Data":"21-03-2018","Ora":"08 : 00 : 00"}
{"Sasi Reshje":0,"Drejtim Ere":"W","Temperatura":17,"Shpejtesi Ere":0,"Data":"21-03-2018","Ora":"08 : 30 : 00"}
{"Sasi Reshje":0,"Drejtim Ere":"W","Temperatura":18,"Shpejtesi Ere":0,"Data":"21-03-2018","Ora":"09 : 00 : 00"}
{"Sasi Reshje":0,"Drejtim Ere":"W","Temperatura":18,"Shpejtesi Ere":1,"Data":"21-03-2018","Ora":"09 : 30 : 00"}
```

PËRFUNDIME

Si konkluzion, vlen për t'u theksuar thjeshtësia që ofron libraria Jersey, në ndërtimin e aplikacioneve RESTful Web Service. Kjo librari e integruar në kod në Java, ofron fleksibilitetin e Jersey dhe pavarësinë e klasave në Java.

Gjithashtu, me interes është afishimi i të dhënave në kod të hapur në format JSON, sepse në këtë mënyrë thjeshtohet aksesimi i tyre direkt nga aplikacione të tjera Web, që mund të jenë biznese rajonale që përdorin apo përpunojnë këto të dhëna.

REFERENCAT

- ALLAMARAJU, S. 2010 : RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity on Yahoo Press; 1 edition, fq 29-45
- LOUVEL, J. , TEMPLIER,T. & Boileau,T. 2012 : Restlet in Action: Developing RESTful web APIs in Java on Manning Publications 1st Edition, fq 81-121
- NEWMAN, S. 2015 : Building Microservices Designing Fine-Grained Systems on O'Reilly Media, fq 71-95
- PARASTATIDIS, S. , WEBBER, J. & ROBINSON, I. 2010 : REST in Practice Hypermedia and Systems Architecture on O'Reilly Media, fq 57-78
- PATNI, S. 2017 : Pro RESTful APIs Design, Build and Integrate with REST, JSON, XML and JAX-RS on APress, fq 36-49
- RICHARDSON, L. , RUBY, S. & AMUNDSEN, M. 2013 : RESTful Web APIs Services for a Changing World on O'Reilly Media, fq 167-215