UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dane Porenta

Razvoj spletnih storitev v Javi

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2012

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dane Porenta

Razvoj spletnih storitev v Javi

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: prof. dr. Matjaž Branko Jurič

Ljubljana, 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00166/2011
Datum: 12.09.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DANE PORENTA**

Naslov: **RAZVOJ SPLETNIH STORITEV V JAVI**
WEB SERVICES DEVELOPMENT IN JAVA

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Analizirajte dve najpogostejši vrsti spletnih storitev: SOAP in REST. Prikažite njihovo delovanje. Opišite in proučite protokol SOAP in opisni jezik WSDL ter specifikacije WS–* in WS–I Basic Profile. Za spletne storitve REST analizirajte arhitekturni stil REST in opisovalno datoteko WADL. Opravite primerjavo in identificirajte prednosti in slabosti uporabe SOAP in REST storitev. Proučite programska vmesnika v Javi – JAX-WS in JAX-RS ter razvijte praktični primer spletne storitve SOAP.

Mentor:

prof. dr. Matjaž B. Jurič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Dane Porenta,

z vpisno številko 63050087,

sem avtor diplomskega dela z naslovom:

Razvoj spletnih storitev v Javi

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  prof. dr. Matjaž Branko Jurič

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela;

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja: _____

Zahvala

Zahvalil bi se prof. dr. Matjažu Branku Juriču za vodenje in pomoč pri izdelavi diplomske naloge. Še posebej bi se zahvalil staršema in bratu za vso podporo v letih študija.

Seznam uporabljenih kratic in simbolov

API – Application programming interface
COM – Component Object Model
DOM – Document Object Model
ebXML – Electronic Business using eXtensible Markup Language
GIF – Graphics Interchange Format
HTML – Hyper Text Markup Language
HTTP – HyperText Transfer Protocol
IP – Internet Protocol
JAXB – Java Architecture for XML Binding
JAXM – Java API for XML Messaging
JAXP – Java API for XML Processing
JAXR – Java API for XML Registries
JAX–RPC – Java API for XML–based RPC
JAX–RS – Java API for RESTful Web Services
JAX–WS – Java API for XML Web Services
JPA – Java Persistence API
JSON – JavaScript Object Notation
MIME – Multipurpose Internet Mail Extension
MSMQ – Microsoft Message Queuing
MTOM – Message Transmission Optimization Mechanism
OASIS – Organization for the Advancement of Structured Information Standards
PKI – Public key infrastructure
POJO – Plain Old Java Object
REST – Representational state transfer
RI – Reference implementation
RMI – Remote Method Invocation
RPC – Remote procedure call
RSS – Really Simple Syndication
SAAJ – SOAP with Attachments API for Java
SAX – Simple API for XML Parsing
SAML – Security Assertions Markup Language
SJSXP – Sun Java Streaming XML Parser
SOAP – Simple Object Access Protocol
SOA – Service Oriented Arhitecture
SSL – Secure Sockets Layer
StAX – Streaming API for for XML
STS – Security Token Service

TCP – Transmission Control Protocol
UDDI – Universal Description, Discovery and Integration
URI – Uniform Resource Identifier
URL – Uniform Resource Locator
URN – Uniform Resource Named
WSDL – Web Service Definition Language
WSI – Web Services Interoperability Organization
W3C – World Wide Web Consortium
WADL – Web Application Description Language
WSIT – Web Services Interoperability Technology
WCF – Windows Communication Foundation
WSE – Web Services Enhancements
XML – Extensible Markup Language
XOP – XML–binary Optimized Packaging
XSD – XML scheme
XSLT – XSL Transformations

Kazalo

Povzetek

V diplomski nalogi smo se osredotočili na spletne storitve in njihov razvoj. Predstavili smo dve najpogostejši vrsti spletnih storitev: SOAP in REST. Pokazali smo njihovo delovanje, razvoj in kako jih odjemalec pokliče. Pri spletnih storitvah SOAP smo podrobno opisali protokol SOAP in opisovalno datoteko WSDL. Opisali smo tudi specifikacije WS–* in WS–I Basic Profile, ki pomaga pri interoperabilnosti. Za spletne storitve REST smo obrazložili arhitekturni stil REST, na katerem slonijo, in opisovalno datoteko WADL. Vključili smo tudi prikaz, v katerem primeru in zakaj je boljše uporabiti spletno storitev REST ali spletno storitev SOAP. V delu razvoja smo opisali vmesnike API v programskem jeziku Java, s katerim lahko razvijamo spletne storitve. Vključena je tudi primerjava med JAX–RS in JAX–WS.

Razvili smo praktični primer spletne storitve SOAP, ki s svojimi funkcijami pregleduje ali spreminja zapise v tabelah v podatkovni bazi. Generirali smo odjemalca za spletno storitev iz datoteke WSDL. Razvili smo tudi primer spletne storitev REST. Primer vključuje podatkovno bazo kot vir sredstev. Določili smo koren za dostop in URI–je za dostop do storitve ter definirali metode za vsako posamezno vrsto zahteve in URI–ja. Na koncu smo razvili še odjemalca REST. Obe storitvi smo razvili z orodjem Netbeans in uporabo strežnika GlassFish, kamor smo ju tudi uspešno namestili in testirali.

Ključne besede:

- spletna storitev,
- SOAP,
- REST,
- WS–*,
- Java,
- vmesnik API.

Abstract

We focused on web services and their development. We presented two of the most common types of web services: SOAP and REST web services. We showed how they work, how to develop and ways how client can invoke them. Looking into SOAP web service details we described it's SOAP protocol and WSDL file for describing web service. We also described WS–* specifications and WS–I Basic Profile, which helps with interoperability. After describing SOAP web services, we described REST web services. Here we explained REST architecture style, which is a backbone for this kind of services, and WADL file to describe service. We also compared SOAP and REST web services, explained why REST service is better and why SOAP service is better and which one is better in different situations. We described API interfaces for development of web services in Java environment. We also compared JAX–RS and JAX–WS.

We developed an example of SOAP web service, which has functions to view or modify records in specified tables in database. In the end, we generated SOAP client for my SOAP web services from its WSDL file. We also developed an example of REST web service. It utilizes database as source of resources. We set access root and URIs for accessing the developed web service and defined methods for each type of requests and URIs. We also developed REST client at the end. We developed both web services with NetBeans tool together with GlassFish server, where we successfully deployed them and have been testing them.

Keywords:

- web service,
- SOAP,
- REST,
- WS–*,
- Java,
- API interface.

1 UVOD

Povezovanje aplikacij in drugih komponent ter izmenjava podatkov med njimi prek spleta je postalo vse bolj pomembno. Spletne storitve predstavljajo enega izmed načinov. Cilj diplomske naloge je analizirati spletne storitve, jih razvrstiti in vsako vrsto teh tudi opisati. Poudarek je na razvoju, kjer opišemo vmesnike API in razvijemo po en primer obeh spletnih storitev.

Na začetku diplomske naloge podrobno opišemo spletne storitve SOAP. Ustavimo se pri protokolu SOAP in pripadajoči datoteki za opis storitve, ki uporablja shemo XSD predstavitve notranjih razmerij med atributi in elementi. Zraven opišemo tudi specifikacije WS–*, ki izboljšajo in dodajo podporo. Ustavimo se tudi pri WS–I Basic Profile, ki pripomore k interoperabilnosti. V tretjem poglavju opišemo spletne storitve REST, arhitekturni stil Representational state transfer in format JSON. Obe vrsti storitvi primerjamo v naslednjem poglavju.

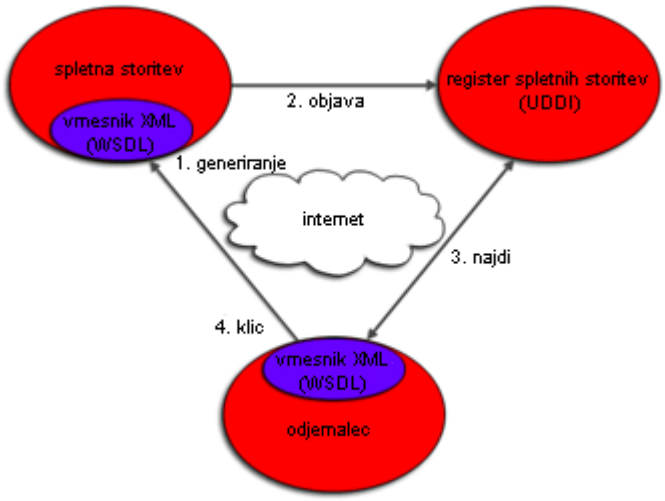
V naslednjemu poglavju opišemo vmesnike API, s katerimi lahko razvijamo spletne aplikacije. Glavna vmesnika v Javi sta JAX–WS, s katerimi razvijamo spletne storitve SOAP, in JAX–RS za razvoj spletnih storitev REST. Opisani so še drugi vmesniki, ki jih lahko uporabljamo posamezno ali dopolnjujejo druge vmesnike.

V zadnjem poglavju opišemo spletni storitvi REST in SOAP, ki smo ju razvili v okviru diplomske naloge. Spletna storitev SOAP s svojimi funkcijami pregleduje ali spreminja zapise v tabelah v podatkovni bazi. Za varnost smo poskrbeli z vmesnikom WSIT. Na koncu smo generirali odjemalca za spletno storitev iz datoteke WSDL. Spletna storitev REST izpostavlja zapise iz podatkovne baze. Izbrali smo sredstva in jih primerno poimenovali z URI–jem. Definirali smo tudi primerne akcije za odgovore na zahteve prek protokola HTTP. Za varnost smo poskrbeli z avtorizacijo prek uporabniškega imena in gesla ter z uporabo protokola SSL. Razvili smo tudi odjemalca.

2 Spletne storitve SOAP

Spletne storitve SOAP z uporabo protokola SOAP prenašajo sporočila, napisana v jeziku XML. Uporabljajo standardne protokole spletne komunikacije, zato so neodvisne od operacijskega sistema in lahko povezujejo aplikacije, ki tečejo na različnih operacijskih sistemih, strojni opremi, različni programski opremi in podatkovnih bazah. Lahko tudi povezujejo več aplikacij iz različnih virov, da se ustvari vtis enotne storitve. Prav tako implementacija spletne storitve ni vidna zunaj spletne storitve.

Slika 1 opisuje potek odkrivanja in klicanja spletnih storitev SOAP. Spletna storitev generira datoteko WSDL in jo objavi v registru UDDI. Potem lahko odjemalec v registru UDDI poišče to ali kakšno drugo registrirano storitev in nazaj dobi dokument WSDL. Iz dokumenta razbere podatke za klic najdene storitve in nato z njo komunicira.



Slika 1: Odkrivanje in klicanje spletnih storitev [1]

2.1 SOAP

SOAP [2] je lahek protokol za izmenjavo strukturiranih podatkov z uporabo tehnologij XML, v katerem se sporočila lahko izmenjujejo prek različnih aplikacijskih protokolov (na primer HTTP in SMPT). Cilja na preprostost in razširljivost.

Slika 2 prikazuje strukturo sporočila SOAP. Sporočilo je dokument XML z naslednjimi elementi:

- Ovojnica
  Določa dokument XML, da je tipa SOAP. Obdaja celo sporočilo.

- Glava

  Je neobvezna in vsebuje dodatne informacije za sporočilo, če jih potrebujemo. Te informacije so, na primer datum, avtorizacija in preusmerjanje. Ima naslednje atribute:
  - atribut actor

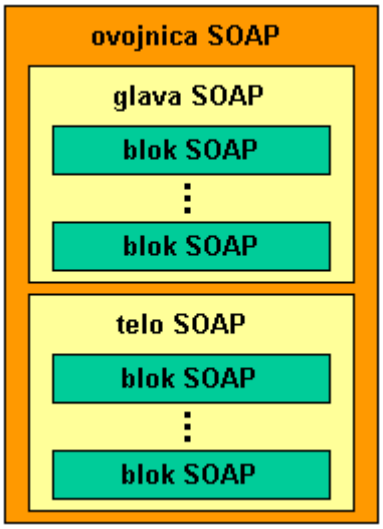
  Z njimi odjemalec določa prejemnika glave SOAP.
  - atribut mustUnderstand

  Označuje, ali je obdelava glave SOAP obvezna ali neobvezna.
- Blok

  Struktura za ločevanje podatkov, ki predstavlja svojo logično enoto.
- Telo

  Vsebina sporočila.
- Napaka

  Je neobvezna. Vsebuje informacije o napakah in/ali statusu sporočila.



Slika 2: Struktura sporočila SOAP [2]

2.2 WSDL

Jezik, razvit na osnovi XML, ki daje format za opis spletne storitve. Odjemalci za dostop do spletne storitve preberejo in interpretirajo datoteko WSDL o lokaciji storitve in njenih operacijah. Je nekakšen vmesnik, ki zagotavlja odjemalcu vse potrebne informacije za uporabo storitve. Odjemalci prek WSDL dobijo informacije, kje je storitev dostopna, njene operacije, komunikacijske protokole in format za pošiljanje sporočil.

2.3 XSD

Shema je abstraktna struktura za diagramsko upodobitev množice podatkov. Določa vrstni red oznak v dokumentu XML, označuje polja, ki so obvezna ali ki se lahko ponovijo, daje podatkovne tipe polje in tako naprej. Shema je pomembna, ker lahko z njo potrdimo, da je dokument veljaven glede elementov, atributov, strukture in tipov podatkov.

XSD [3] je trenutni standard za vse dokumente XML in podatke. Omogoča definicijo strukture, elementov, podatkovnih tipov in atributov za vse dokumente XML v skladu z organizacijo W3C. Poleg lastnih podatkovnih tipov (kot so celo število, niz in tako naprej) sheme XSD omogočajo tudi opredelitev novih podatkovnih tipov z uporabo elementov <simpleType> in <complexType>.

Na sliki 3 so vidni elementi sheme XML:

- <XSD Schema>
  Najvišji element na ravni sheme, vsebovati mora imenski prostor http://www.w3.org/2001/XMLSchema.

- <Attribute>
  Atributi so lahko definirani kot otroci v <Element> ali znotraj <ComplexType>. Določimo lahko tip, omejujemo obseg vrednosti in drugo.

- <Element>
  Ta element je v korenu vsake sheme. Definira entiteto.

- <ComplexType>
  Definira tip elementa, ki lahko vključuje več elementov <Attribute> in <Element>.

- <SimpleType>
  Definicija tipa za vrednosti, ki se lahko uporabljajo kot vsebine v <Element> ali <Attribute>. Ne more vsebovati <Elements> ali imeti <Attribute>.

- <Sequence>
  Elementi v <Sequence> se pojavijo v zaporedju. Vsak se lahko pojavi od nič do poljubnokrat.

- <Choice>
  Omogoča izbiro enega elementa od definiranih elementov v <Choice>.

- <All>
  Dovoljuje, da se nekateri ali vsi elementi v skupini pokažejo v poljubnem zaporedju.

- <Restriction>
  Pri preprosti vsebini omejuje obseg vrednosti na podmnožico podedovane preproste vrste tipov. Pri kompleksni vsebini pa omejuje vsebino podedovanega kompleksnega tipa.

- <Union>
  Združi več elementov <SimpleType> v en element <SimpleType>.

- <List>
  Definira element <SimpleType> kot seznam določenega podatkovnega tipa.

- <Facet>
  Je značilnost podatkovnega tipa, s katero lahko omejimo dovoljene vrednosti.

- <Any>
  Razširimo dokument XML z novimi elementi. Novi elementi se pojavijo iz določenega imenskega prostora v elementu <ComplexType>.

- <AnyAttribute>
  Razširimo dokument XML z novimi atributi. Novi <Attribute> se pojavijo iz določenega imenskega prostora v elementu <ComplexType>.

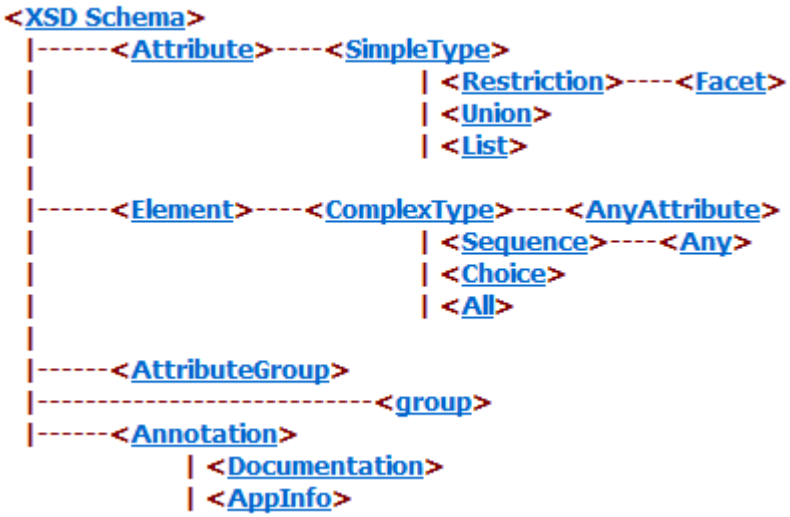
- <Group>
  Skupina deklaracij elementov, da se lahko vključijo v element <ComplexType> kot skupina.

- <AttributeGroup>
  Skupina deklaracij atributov, da se lahko vključijo v element <ComplexType>.

- <Annotation>
  Uporablja se za hranjenje dodatnih informacij o shemi ali elementih. Lahko vsebuje enega ali več primerov <Documentation> ali <AppInfo>.

- <Documentation>
  Specifične informacije, ki se ne uporabljajo za potrjevanje, na primer komentarji.

- <AppInfo>
  Definira informacije, povezane z aplikacijo.



Slika 3: Seznam vseh elementov XSD sheme [3]

2.4 UDDI

UDDI [4] je register za spletne storitve in ne podatkovna baza storitev. S povezavami nas napoti k iskani storitvi. Iz slike 4 je razvidno, kako UDDI deluje. Razvijalci spletnih storitev in podjetja lahko objavijo in oglašujejo svoje spletne storitve. Lahko pa tudi najdejo storitev, ki ustreza njihovim zahtevam, ali najdejo iskano storitev z aplikacijo.

Informacije o storitvah so za programerska podjetja in razvijalce shranjene v Service Type ali tModel. Ta opisuje na primer tip spletne storitve, uporabljene protokole in drugo.

Informacije o storitvi za podjetja sestavljajo tri komponente:

- bele strani s podatki o podjetju, kot so telefonska številka, elektronska pošta, opis poslovanja;
- rumene strani, ki jih razvrstijo glede na kategorijo poslovanja in tip storitve;
- zelene strani s tehničnimi podatki o spletni storitvi.

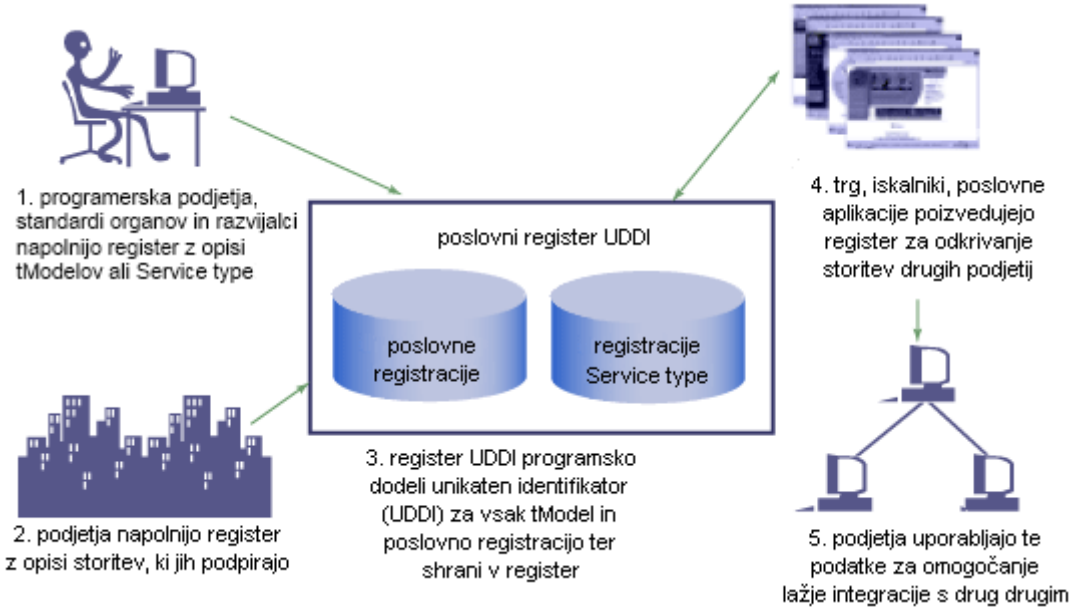
Vmesnik UDDI vsebuje metode za odkrivanje in pridobivanje potrebnih podatkov od storitve. Iskanje je mogoče po imenu, kategoriji, tModelu, identifikatorju in URL–ju.

Ima tri tipe registrov:

- Zasebni
  Notranji register, postavljen za požarnim zidom in izoliran od javnega omrežja. Administrativne pravice in podatki so nedostopni. Ni skupne uporabe.
- Povezani
  Register je v kontroliranem okolju, samo pooblaščeni odjemalci imajo dostop. Podatki so lahko v skupni rabi, ampak kontrolirano.

- Javni
  Javno odprti podatki so lahko v skupni rabi ali se prenašajo med seboj, vsebina je lahko nadzorovana.

UDDI za uporabo zahteva XML, HTTP, WDSL in SOAP. Uradna spletna stran je www.uddi.org.



Slika 4: Prikaz delovanja UDDI–ja [4]

2.5 Specifikacije WS–*

Specifikacije za spletne storitve so pod razvojem Oasisa, Microsofta, IBM–a in drugih podjetij ali posameznikov. Namen specifikacij je standardizirati spletne storitve glede varnosti, sporočanja in transakcij. Specifikacije razdelimo v skupine, kot je vidno na sliki 5.

Slika 5: Skupine specifikacij [5]

2.5.1 Skupina XML

2.5.1.1 XML

Extensible Markup Language [9] je preprost in razširljiv označevalni jezik. Ni programski jezik, ampak prenaša in opisuje strukturirane podatke, zato je primeren za izmenjavo podatkov. Po definiciji je dokument XML niz znakov. Lahko se pojavi vsak znak UNICODE v dokumentu. Podatek je opisan tako, da ga obdamo z obeh strani z oznako (angl. tag). Oznake so prilagodljive in dajo informacijo o podatkih in strukturi dokumenta. XML je neodvisen in človeško berljiv dokument.

2.5.1.2 Imenski prostori v XML

So [10] uporabljeni za unikatno imenovanje elementov in atributov v dokumentih XML. Zaradi enakih imen elementov lahko pride do dvoumnosti. To se reši tako, da se doda predpono pred imenom ali uporabi atribut xmlns, ki mora vsebovati URI (angl. Uniform Resource Identifier) za identificiranje internetnega vira. Najpogostejša oblika URI–ja je URL ali URN.

2.5.1.3 Nabor informacij v XML

Abstraktni nabor podatkov. Namen je zagotoviti nabor definicij za uporabo v drugih specifikacijah, ki se morajo nanašati na informacije v dobro oblikovanem dokumentu XML. Nabor informacij [11] je lahko ustvarjen z metodami z izjemo razčlenjevanja dokumenta. Sestavljen je iz številnih informacijskih predmetov, ki opisujejo neki del dokumenta. Vsak informacijski element ima lastnosti.

2.5.2 Skupina Metapodatki

2.5.2.1 WSDL

2.5.2.1.1 WSDL 1.1

Nadaljevanje opisa iz poglavja 2.2. WSDL [7] uporabi sintakso jezika XML za opis spletnih storitev kot zbirko končnih točk, zmožnih izmenjave sporočil. Za definiranje spletne storitve WSDL uporabi šest elementov:

- port type: skupine in opis operacij storitve;
- port: vrata za povezavo;
- message: imena in format sporočil, ki ga storitev podpira;
- type: opiše podatkovne tipe, uporabljene v sporočilih;
- binding: komunikacijski protokoli;
- service: podaja naslov URL za dostop.

Elementi message, port type,operation in type so abstraktne definicije, elementa binding in service pa sta konkretni definiciji. Abstraktne in konkretne definicije so ločene, kar omogoči ponovno uporabo abstraktnih definicij.

Dodatno imamo še tri elemente:

– Definition

  Je korenski element in pripada imenskemu prostoru http://schemas.xmlsoap.org/wsdl/.

– Import

  Omogoča uvoz dokumentov.

– Document

  Omogoča pisanje komentarjev ali dokumentacije.

2.5.2.1.2 WSDL 2.0

Obstaja tudi različica 2.0 [13], ki prenese kar nekaj sprememb in novosti:

- element definition je preimenovan v description,
- element port je preimenovan v endpoint,
- odstranitev elementa message,
- element service lahko implementira en vmesnik samo prek atributa interface,
- element portType je preimenovan v interface ter dobi nove elemente in atribute:
  – atribut extends
  Dovoljuje enemu vmesniku, da razširi ali podeduje od enega ali več vmesnikov.
  – atribut styleDefault
  Definira pravila za operacije.
  – atribut pattern
  Dovoljuje uvoz MEP, ki je definirana drugje, v definicijo operation.
  – atribut messageLabel
  Prikaže vlogo sporočila v MEP.
  – element fault
  Opiše napako z abstraktno vsebino, ki se mogoče zgodi.
  – elementa infault in outfault
  Prek teh se lahko operacije sklicujejo na element fault z atributom ref.
- vpelje novi globalni atribut wsdlx:safe, ki prikaže ali je operacija varna ali ne. Varna pomeni, da je samo bralna operacija ali operacija odjemalcu ne daje novih obveznosti.

2.5.2.2 WSDL Binding Extension for SOAP 1.2

Opredeljuje [12] razširitve za WSDL 1.1, s katerimi pokažemo, da sporočila spletne storitve uporabljajo protokol SOAP 1.2.

2.5.2.3 WS–Policy

WSDL opisuje operacije in sporočila, ponujene v spletni storitvi, ne daje pa informacij o zahtevah, ki so potrebne, da se spletna storitev (po)kliče. To opiše specifikacija WS–

Policy [14] ter dodatno še storitvene splošne značilnosti in zmogljivosti. Ne določa pa, kako se ta pravila odkrijejo, ali kako so pritrjena v spletno storitev.

2.5.2.4 WS–PolicyAttachment

Definira dva splošna mehanizma [15] za povezovanje pravil in objektov, na katerih se pravila uporabljajo. Pravila so lahko definirana kot del obstoječih metapodatkov o objektu ali definirana neodvisno in povezana prek zunanje povezave. Prvi mehanizem dovoljuje elementom XML, da se poveže s pravilom kot del njihovih definicij. Drugi mehanizem dovoljuje pravilom, da so povezani s pravili objektov, neodvisno od njihovih definicij.

2.5.2.5 WS–MetadataExchange

Spletne storitve uporabljajo metapodatke, da opišejo, kaj potrebujejo druge končne točke, da se lahko sporazumevajo. To naredijo z uporabo mehanizmov WS–MetadataExhange [16], s katerimi se pridobivajo trije tipi metapodatkov: WSDL, WS–Policy in shemo XML. Mehanizmi dobijo metapodatke z metodo GET.

2.5.2.6 WS–Discovery

Protokol za odkrivanje storitev [17]. Storitve odkriva tako, da pošlje sporočilo ali informacijo vsem prejemnikom v ciljni skupini hkrati in v enem prenosu. Kreira kopije, če vmesna točka povezave to zahteva. Storitve se lahko odkrivajo po tipu, obsegu ali oboje. Odjemalec pošlje sondo ciljni skupini, in storitve, ki ustrezajo sondi, pošljejo odgovor direktno odjemalcu. Lahko se išče tudi po imenu. Odjemalec pošlje zahtevo ciljni skupini in prava storitev pošlje odgovor odjemalcu.

2.5.2.7 WS–MTOMPolicy

Specifikacija [18] za opisovanje pravil za domene za trditve MTOM, ki so lahko specificirane v pravilih, opisanih v WS–Policy.

2.5.3 Skupina Sporočanje

2.5.3.1 SOAP

Nadaljevanje opisa iz poglavja 2.1. SOAP je v temelju enosmerna izmenjava brez stanja med vmesnimi točkami SOAP, od pošiljatelja do prejemnika. Aplikacije lahko tvorijo bolj zahtevne interakcije z združevanji enosmernih izmenjav, kot sta interakcija zahteva–odgovor ali zahteva–več odgovorov. SOAP prikrije vse podatke v povezavi z usmerjanjem sporočila, prenosom zanesljivih podatkov, prečkanjem požarnega zidu in drugo.

SOAP zagotavlja model za porazdeljeno obdelavo sporočil in privzame, da sporočila potujejo od pošiljatelja do prejemnika z nič ali več posredniki. Vsaka od teh vmesnih točk obdela sporočilo na svoj način. Podpira tudi številne vzorce za izmenjavo

sporočil (angl. Message Exchange Patterns). Model določa vmesne točke z naslednjimi atributi:

- Atribut role
  Določi tip vmesne točke: pošiljatelj, sprejemnik ali posrednik.
- Atribut mustUnderstand
  Določa nadaljnje akcije za obdelavo sporočila.
- Atribut relay
  Določa, če mora biti glava, ki cilja na posrednika, zamenjana pod pogojem, da ni obdelana.

2.5.3.2 WS–Addressing

WS–Addressing [19] zagotavlja transportno–nevtralne mehanizme, ki spletnim storitvam dovoljujejo prenašanje informacij za naslavljanje spletnih storitev in sporočil. Določa element XML v glavi SOAP, da se identificirajo končne točke spletne storitve in zagotovitev varnosti v sporočilih.

2.5.3.3 MTOM

Metoda za učinkovito pošiljanje binarnih podatkov. Po navadi so binarni podatki v sporočilu SOAP kodirani v tekst. Velikost binarnih podatkov se poveča za 33 odstotkov. Z MTOM [20] se lahko pošlje binarne podatke kot binarne podatke brez kodiranja v tekst.

2.5.3.4 WS–Enumeration

Za številne aplikacije ena zahteva in en odgovor ne zadovoljujeta za prenašanje velikih podatkov. Specifikacija [21] definira protokol SOAP za oštevilčevanje, ki dovoljuje podatkovnim sredstvom, da zagotovi abstraktno sejo odjemalcu, imenovano oštevilčena vsebina (angl. enumeration context). Predstavlja nekakšen kazalnik skozi nize podatkov. Odjemalec lahko potem zahteva elemente XML skozi oštevilčeno vsebino nad enim ali več sporočili SOAP. Stanje iteracije mora biti shranjeno med zahtevami od strani podatkovnega sredstva ali odjemalca.

2.5.3.5 WS–Eventing

Spletne storitve pogosto želijo prejemati sporočila, ko se zgodi neki dogodek v drugi aplikaciji ali spletni storitvi. Ta specifikacija [22] opisuje mehanizme, ki to omogočajo. Spletna storitev je lahko izvir dogodkov ali naročnik. Naročnik se zanima in naroči za sprejemanje sporočil o dogodkih, ki se zgodijo v drugi spletni storitvi, ki je izvir dogodkov. Naročnik lahko upravlja naročnino z upravljalnikom naročnine. Zagotavlja možnost za naročnika, da obnovi ali prekliče naročnino.

2.5.3.6 WS–Transfer

Opisuje [23] mehanizme za pridobivanje predstavitev entitet XML z uporabo arhitekture spletnih storitev. Definira dva tipa entitet:

– sredstva, katerih entitete so naslovljena s povezavo končne točke, ki zagotovijo predstavitev XML;
– dejavniki sredstev, ki so spletne storitve in lahko ustvarijo novo sredstvo iz predstavitve XML.

Bolj natančno opisuje dve operaciji za pošiljanje in prejemanje predstavitve ter dve operaciji za ustvarjanje in brisanje sredstva in njegove predstavitve.

2.5.3.7 SOAP–over–UDP

Ta specifikacija [24] omogoča transport sporočil SOAP prek protokola UDP z zanesljivo dostavo. Ker protokol UDP sam po sebi ne zagotovi zanesljive dostave, specifikacija doseže zanesljivo dostavo, tako da pošljejo sporočilo ali informacijo ciljni skupini hkrati in v enem prenosu. Kreira kopije, če vmesna točka povezave to zahteva. Je nujna za storitve, ki uporabljajo WS–Discovery.

2.5.3.8 SOAP Binding for MTOM

Natančneje opisuje [25] spremembe SOAP MTOM in XML–Binary Optimized Packaging, potrebne za njihovo uporabo s protokolom SOAP.

2.5.4 Skupina Transakcije

2.5.4.1 WS–Coordination

Specifikacija [26], ki zagotavlja širok nabor usklajevalnih protokolov za usklajevanje dejavnosti med udeleženci. Omogoča sporazumni dogovor o izidu porazdeljenih dejavnosti. Izvedba slednje pogosto traja dolgo zaradi poslovnega odlašanja in uporabniške interakcije.

2.5.4.2 WS–AtomicTransaction

Definira [27] usklajevalne tipe za atomske transakcije. To so transakcije, ki se zgodijo ali ne. Tipi usklajujejo dejavnosti, ki imajo lastnosti vse ali nič. Definira protokole, ki omogočajo obstoječim sistemom za obdelovanje transakcij, da ovijejo njihove protokole in sodelujejo prek različnih proizvajalcev strojne in programske opreme.

2.5.4.4 WS–BusinessActivity

Definira [28] usklajevalne tipe za poslovne dejavnosti, ki usklajujejo dejavnosti. Te dejavnosti dajejo poslovno logiko za obravnavanje izjem, ki se zgodijo med izvedbo dejavnosti poslovnega procesa. Akcije se zgodijo takoj in so dokončne, lahko so poklicane v dogodku napake. Definira protokole, ki omogočajo obstoječim poslovnim procesom in delovnim potekom, da ovijejo njihove mehanizme in sodelujejo čez meje zaupanja in različne implementacije prodajalcev.

2.5.5 Skupina Zanesljivo sporočanje

2.5.5.1 WS–ReliableMessaging

Pogosto je pogoj za dve spletni storitvi, ki želita komunicirati, da to storita zanesljivo brez prisotnosti programske, sistemske ali omrežne napake. Ta specifikacija [29] ustvari mehanizem za zanesljiv prihod sporočila. Identificira, sledi in upravlja prihod sporočila med virom in ciljem. Natančneje tudi opisuje tudi spremembe za uporabo s SOAP. Dovoljuje razširljivost z drugimi specifikacijam, recimo WS–Security ali WS–Policy.

2.5.5.2 WS–RM policy

Definira [30] pravila za domenske trditve, namen je zagotoviti zanesljiv prenos. Pomaga si z WS–Policy in WS–ReliableMessaging.

2.5.6 Skupina Varnost

2.5.6.1 WS–Security: SOAP Message Security

Ta specifikacija [31] predlaga standardni nabor razširitev SOAP, ki jih lahko uporabljamo, ko razvijamo varno spletno storitev. S tem poskušamo zagotoviti integriteto sporočila in zaupnost. Vključuje PKI, Kerberos, SSL, SAML in podporo raznim varnostnim žetonom, zaupnim domenam, podpisovanju in šifriranju. Zagotavlja tri mehanizme: pošiljanje varnostnih žetonov kot del sporočila, integriteto in zaupnost sporočila. Pri sporočilu SOAP se doda varnostna glava kot del sporočila, v kateri so podatki o kriptografiji ali podpisu.

2.5.6.2 WS–Security: UsernameToken Profile

Opisuje [32], kako uporabiti žeton z uporabniškim imenom (angl. username token) z WS–Security: SOAP Message Security. Bolj natančno opiše, kako lahko odjemalec dobavi žeton z uporabniškim imenom kot sredstvo identifikacije pošiljatelja z uporabniškim imenom in po možnosti z geslom.

2.5.6.3 WS–Security: X.509 Certificate Token Profile

Specifikacija [33], ki opisuje uporabo certifikata X.509 z WS–Security: SOAP Message Security. Certifikat X.509 opisuje povezavo med javnim ključem in naborom atributov, ki vključuje ime objekta, ime izdajatelja, serijsko številko in veljavni čas. Lahko se uporabi za preverjanje javnega, ki se uporabi za preverjanje pristnosti sporočila SOAP ali identifikacijo javnega ključa iz šifriranega sporočila.

2.5.6.4 WS–SecurityPolicy

Specifikacija [34] omogoča, da spletne storitve izrazijo svoje varnostne omejitve in zahteve, ki so glede žetonov, šifrirnih algoritmov in mehanizmov, vključno z varnostjo na transportni plasti. Namen je zagotoviti dovolj informacij za povezljivost in združljivost določene strani udeležencev in vse informacije, potrebne za komunikacijo, da se omogoči varna izmenjava sporočil.

2.5.6.5 WS–Trust

WS–Trust [35] razširja WS–Security z metodami za izdajo, obnovo in potrjevanjem žetonov ter načine za vzpostavitev in oceno zaupanja in prisotnosti v varni izmenjavi sporočila. WS–Security predvideva, da se žeton ustvari zunaj spletne aplikacije in uporabo tretje varnostne tehnologije. WS–Trust pa omogoča modeliranje operacij za pridobivanje žetonov. WS–Trust definira entitete in sporočila za izdajo žetonov čez spletno storitev STS.

2.5.6.6 WS–SecureConversation

Definira [36] razširitve za WS–Security, ki dovoljuje varnostni vsebini vzpostavitev in delitev ter izpeljave sejnih ključev. Posledično se izboljša splošna učinkovitost in varnost dodatnih izmenjav. Varnostna vsebina se definira kot nov žeton WS–Secury.

2.5.6.7 WS–Federation

Definira mehanizme [37], ki dovoljujejo različnim varnostnim skupinam, da se združijo v federacijo. Združijo se na tak način, da je pooblaščeni dostop do sredstev opisan v eni varnostni skupini lahko posredovan drugimi varnostnimi skupinami, katerih identiteta se upravlja v drugih območjih. Federacija je zbirka dveh ali več entitet, kjer so sredstva ene entitete dostopna drugi entiteti. Federacije posredujejo identitete, atribute, pooblastila in dovoljenja. Zgrajena je na WS–Trust in WS–Security.

2.5.6.8 WS–Federation Active Requestor Profile

WS–Fedration Active Requestor Profile [38] definira, kako aktivni pošiljatelji zahtev uporabljajo federacije, opisane v WS–Federation, na primer aplikacije SOAP.

2.5.6.9 WS–Federation Passive Requestor Profile

Specifikacija [39] definira, kako pasivni pošiljatelji zahtev uporabljajo federacije, opisane v WS–Federation, na primer spletni brskalniki, ki podpirajo HTTP.

2.5.6.10 WS–Security: Kerberos Binding

Kerberos je varnostni protokol za preverjanje pristnosti. Ta specifikacija [40] opisuje integracijo Kerberosa v varnostno arhitekturo spletne storitve.

2.5.6.11 Web Single Sign–On Interoperability Profile

Opisuje skupino protokolov [41] za uporabo z Web Single Sign–On Metadata Exchange Protocolom.

2.5.6.12 Web Single Sign–On Metadata Exchange Protocol

Ponuja mehanizme [42], kjer ciljne storitve lahko določajo protokol, ki ga podpira odjemalčeva identiteta, in uporabo protokolov, ki so podprti, za dodatno komunikacijo z odjemalčevo identiteto. Definira nevtralne mehanizme za določitev podprtih protokolov, s tem dovoljuje storitvi, da določi pravi protokol za uporabo za sprožitev

obdelave identitete. Prav tako definira proces za določanje identiteti ponudnika za danega odjemalca.

2.6 Interoperabilnost

Pomemben dejavnik pri spletnih storitvah je interoperabilnost, ki pomeni povezljivost storitev prek različnih platform, operacijskih sistemov in programskih jezikov. Za spletne storitve obstaja veliko specifikacij in protokolov, vendar jo ne omogočajo. Za to skrbi organizacija WS–I (Web Services Interoperability Organization), tako da vzpostavi pravila uporabe izbranih specifikacij in protokolov. WS–I ne piše specifikacij ali protokolov, ampak tesno sodeluje z organizacijami, ki jih razvijajo. Organizacija WS–I ponuja vire [72] za razvijalce, da ustvarijo spletne storitve po pravilih organizacije:

– Profili

Dajejo napotke, kako naj bodo specifikacije uporabljene skupaj za najboljšo interoperabilnost.

– Vzorčne aplikacije

Ponuja vzorčne storitve, ki so v skladu s pravili WS–I in pomagajo razvijalcem pri razvoju.

– Orodja za testiranje

Uporabljeni za testiranje, če so poslana sporočila v skladu s Profili. Trenutno je testiranje mogoče samo za WS–I Basic Profile.

2.6.1 WS–I Basic Profile

Je [43] nabor specifikacij za interoperabilnost. Natančno opiše, kako se SOAP, HTTP, XML, WSDL in UDDI uporabijo skupaj za doseganje interoperabilnosti in kdaj je primerno uporabiti kateri protokol, standard ali katero komponento. Tako ustvari smernice za uporabo omenjenih specifikacij in določa strukturo opisnih dokumentov in izmenjanih sporočil ter njihove elemente. Določa dodatne omejitve in definira manjši nabor veljavnih storitev kot samo SOAP ali WSDL.

Upoštevana vodilna načela:

- Interoperabilnost ni zajamčena.

  Nemogoče je stoodstotno jamčiti interoperabilnost določenih storitev. Basic Profile naslavlja najpogostejše probleme, najdene do zdaj.

- Aplikacijske semantike

  Komunikacijo aplikacijskih semantik se lahko olajša s tehnologijami, ki obsegajo Basic Profile. Slednji zagotavlja, da osnovno razumevanje teh semantik ni naslovljeno v njemu.

- Možnost testiranja
  Ko je mogoče, Basic Profile naredi trditve, ki se jih dá preizkusiti, čeprav to ni potrebno. Boljše je doseči testiranje na nevsiljiv način.

- Moč zahtev
  Basic Profile dela močne zahteve, kjer je le mogoče. Če tega ne more, naredi pogojne zahteve. Izbirne zahteve privedejo nejasnosti in neusklajenosti med implementacijami.

- Omejitve in sprostitve
  Ko razširjamo zahteve določeni specifikaciji, jih lahko Basic Profile omeji in ne sprosti.

- Več mehanizmov
  Če določena specifikacija dovoli uporabo več mehanizmov izmenično, se izbere samo tiste, ki so dobro razumljeni, uporabni in širše implementirani. Tuji ali nerazumljeni mehanizmi prinesejo kompleksnost in zmanjšajo interoperabilnost.

- Kompatibilnost v prihodnosti
  Basic Profile uravnava svoje zahteve z revizijami za naslovljene specifikacije, če je mogoče. To omogoča eleganten prehod in zagotavlja, da se WS–I ne loči od teh prizadevanj. Če ne more nasloviti problema, informacijo prenese pravemu naslovniku.

- Kompatibilnost z razporejenimi storitvami
  Kompatibilnost z novimi storitvami, ki delujejo s starimi vhodi in podatki, ni cilj Basic Profila. Ne uvede novih sprememb omejitev, razen če tako ne reši probleme interoperabilnosti.

- Osredotočenost na povezljivost
  Osredotoča se samo na nedoslednosti in pomanjkljivosti v oblikovanju specifikacije, ki naslavljajo povezljivost.

- Skladnost ciljev
  Kjer je mogoče, se dá omejitve raje na artefakte (angl. artifacts), kot da bi uporabili vloge ali obnašanje programske opreme. Artefakte se lažje preveri, skladnost je lažje razumljiva in manj možnosti je za napako. Artefakt je lahko na primer opis WSDL ali sporočilo SOAP.

- Povezljivost na nižjih slojih

  Basic Profile govori o povezljivosti na aplikacijskem sloju in privzame, da je povezljivost na nižjih slojih (TCP, IP, Ethernet) dobro razumljiva. Podobno so trditve o protokolih na aplikacijskem sloju (SSL, HTTP) narejene samo, ko problem vpliva na spletne storitve. WS–I ne poskuša zagotoviti povezljivosti teh protokolov kot celoto.

3 Spletne storitve REST

Spletne storitve REST slonijo na REST z uporabo protokola HTTP. REST [44] je arhitekturni stil in ne protokol, ki določa skupino arhitekturnih omejitev, ki omejujejo vlogo in funkcije arhitekturnih elementov in medsebojne povezave. Popolno izkoristi zmogljivost spleta in uporabo njegovih standardov ter z omejitvami omogoči modeliranje ciljne arhitekture. Sestavljen je iz odjemalcev in strežnikov. Odjemalec pošlje zahtevo, ko je pripravljen na prehod na novo stanje, strežnik te zahteve obdela in pošlje nazaj pravilne odgovore. Zahteve in odgovori so zgrajeni okoli predstavitev sredstev (angl. representations of resources). Spletne storitve REST nimajo predpisanih protokolov in standardov, vseeno pa uporabljamo HTTP, URL, XML, HTML, GIF in MIME tipe. Nismo vezani na uporabo določenih razvojnih okolij in protokolov.

3.1 Arhitekturni stil REST

REST ima šest arhitekturnih omejitev [44], vendar omogoča svobodno implementacijo svojih komponent:

- Odjemalec–strežnik
  Odjemalec in strežnik sta ločena. Odjemalca ne zanima podatkovna baza, to je pomembno samo za strežnik. Na drugi strani strežnike ne zanima uporabniški vmesnik ali stanje uporabnika. Tako so lahko strežnik in odjemalci razviti neodvisno. Odjemalci so prenosljivi in strežniki bolj preprosti in razširljivi.

- Brez stanja
  Med prenosi odjemalec–strežnik nobeni odjemalčevi podatki niso shranjeni na strežniku. Vsaka zahteva od odjemalca vsebuje vse potrebne informacije za izvršitev zahteve in vsaka seja je shranjena na odjemalcu. Strežnik lahko prepozna stanja, tako da je stanje na strežniku dosegljivo prek URL–ja. Strežniki so tako bolj vidni za spremljanje in bolj prilagodljivi.

- Predpomnilnik
  Odjemalci lahko shranijo odgovore v predpomnilnik. Odgovori morajo zato biti definirani, ali se ga zapomni ali ne, da se prepreči odjemalcu shranjevanje starih in neprimernih podatkov.

- Več slojev
  Odjemalec je lahko povezan s končnim strežnikom direktno ali prek posredniških strežnikov. Posredni strežniki lahko izboljšajo razširljivost na način, da uravnovesijo obremenitev in omogočijo skupni predpomnilnik.

- Koda na zahtevo
  Dodatna neobvezna omejitev. Strežnik lahko začasno spremeni ali doda funkcionalnost odjemalcu z izvajanjem kode v obliki skript in appletov.

- Enotni vmesnik
  Enotni vmesnik med odjemalcem in strežnikom, ki sledi petim načelom. Poenostavi arhitekturo in loči implementacijo in storitev, kar omogoči neodvisen razvoj. Ta načela so:

  1. Določitev sredstev
     Sredstva so identificirana v zahtevah in se prenašajo med odjemalcem in strežnikom. Vsaka informacija je lahko sredstvo. Sredstvo je konceptualna mapa nabora entitet in ne entiteta sama.

  2. Predstavitev sredstev
     Predstavitev je trenutno ali zaželeno stanje sredstva. Eno sredstvo ima lahko več predstavitev. Po navadi so v formatih HTML, XML in JSON.

  3. Samo–opisna sporočila
     Vsaka zahteva ali odgovor vsebuje svoj opis sheme. Opisani so s shemo XML in z imenskim prostorom XML. Lahko vsebuje tudi metapodatke.

  4. Hypermedia kot izbiralec stanj aplikacije
     Hypermedia je logična razširitev izraza hypertext, v kateri se grafike, avdio, video, golo besedilo in hiperpovezave prepletajo, da se ustvari nelinearni tok informacij. Hypermedia omejuje možnosti odjemalca in odločitve odjemalca vplivajo na stanje aplikacije. Slednje nam pove, kako daleč je uporabnik pri dokončanju opravila. Vsako stanje omogoča samo nekaj odločitev za prehod v novo stanje. V vsakem stanju so opisane povezave, odjemalec se odloči za primer povezave na podlagi metapodatkov povezav. Odjemalec mora poznati samo začetni URL.

  5. Enotne operacije
     Odjemalec komunicira z zaporednim pošiljanjem enega od ukazov POST, GET, PUT, DELETE na samo–opisnih sredstvih. Za to mora razumeti sheme XML in semantiko relacijskih tipov.

REST arhitekturni stil cilja na:

– razširljivost za interakcije komponent;
– posplošitev vmesnikov;
– neodvisen razvoj komponent;
– vmesne komponente za zmanjšanje časovnih zamikov, uveljavljanje varnosti in enkapsulacije starejših komponent, še zmeraj potrebnih za uporabo.

3.2 WADL

WADL [45] je primeren za opisovanje spletnih aplikacij, ki uporabljajo protokol HTTP in XML. Te aplikacije so tipične spletne storitve REST. WADL opiše spletno storitev REST, torej podobno kot WSDL pri spletnih storitvah SOAP. Opisuje spletne storitve REST z naborom elementov <resource>, kjer vsak vsebuje elemente <param> za opisovanje parametrov ter elemente <method> za opisovanje zahteve in odgovora posameznega sredstva.

3.3 Format JSON

JavaScript Object Notation [46] je preprost format za izmenjavo podatkov. Je enostaven za branje, pisanje in tudi za računalniško branje in generiranje. Temelji na podmnožici programskega jezika JavaScript za predstavljanje podatkovnih struktur in objektov. Kljub povezavi z jezikom JavaScript je neodvisen od programskega jezika in ima dostopne razčlenjevalnike v večini programskih jezikih. Pogosto je uporabljen za izmenjavo podatkov med strežnikom in odjemalcem kot alternativa jeziku XML.

JSON temelji na dveh strukturah:

- Zbirka parov ime/vrednost. V različnih jezikih je realizirana kot objekt, zapis, struktura, slovar, razpršena tabela ali asociativno polje.
- Urejen seznam vrednosti. V večini jezikov je realiziran kot polje, vektor, seznam ali zaporedje.

To sta univerzalni podatkovni strukturi. Vsi moderni programski jeziki ju poznajo v taki ali drugačni obliki. Smiselno je, da tudi jezik za izmenjavo podatkov, ki je neodvisen od programskega jezika, temelji na teh dveh strukturah.

4 Primerjava spletnih storitev SOAP in REST

Glavne razlike med SOAP in REST[47, 48]

REST	SOAP
Protokol HTTP	Protokoli HTTP, TCP in drugi
Izpostavlja sredstva, ki predstavljajo podatke.	Izpostavlja operacije, ki predstavljajo logiko.
Ukazi HTTP: GET,POST,DELETE,PUT	Ukaz HTTP POST
Poudarek na točka–točka komunikaciji prek HTTP–ja	Poudarek na ohlapno povezanih sistemih, porazdeljeno sporočanje
Podpira več podatkovnih tipov.	Podpira samo format XML.
Poudarek na komunikaciji brez stanja	Podpira operacije z in brez stanja.
Samo sinhrono sporočanje	Asinhrono in sinhrono sporočanje
Manj omejitev podatkovnih tipov (angl. weak typing)	Več omejitev pri podatkovnih tipih (angl. strong typing)
Lahko si zapomni odgovore HTTP GET, ki jih pozneje uporabi.	Ni podpore za shranjevanje odgovorov.

Tabela 1: Razlike med SOAP in REST

Spletne storitve REST so boljše, ker:

- jih lahko uporablja vsak odjemalec, celo spletni brskalnik z Ajaxom in JavaScriptom;
- so lahkotne, ker ne potrebujejo razčlenjevanja XML in porabijo manj pasovne širine, ker ne potrebujejo glave SOAP za vsako sporočilo;
- so enostavnejše za uporabo in učenje;
- so varne, ker lahko izločajo zahteve, ki niso bralne GET.

Spletne storitve SOAP so boljše, ker:

- imamo za razvoj storitev SOAP na voljo veliko podporo. Pišemo lahko svoje tipe in kode za izjeme. Razvoj odjemalca REST je lahko včasih zahteven, ker moramo pisati surove ukaze in pogledati odgovore HTTP–ja.
- podpirajo asinhrono in sinhrono komunikacijo, spletne storitve REST pa samo sinhrono komunikacijo.
- so bolj varne, ker lahko uporabljajo podporo za pridobivanje varnostnih žetonov v specifikacijah WS–*. REST nima te podpore in ogroža varnost s parametri, ki so lahko del URI–ja.
- imajo podporo za zanesljivo pošiljanje v specifikacijah WS–*. Spletne storitve REST pa niso zanesljive, neuspešne dostave se ponovno pošilja.
- so lažje za upravljanje, ker imajo več omejitev podatkovnih tipov.
- imajo register. Vidimo, kdo jih uporablja, ali kako jih odkrijemo. Storitve REST nimajo registra.

Spletne storitve REST so boljše za spletne storitve. So boljša izbira tudi, če imamo omejeno pasovno širino ali omejene vire. Boljše so za izpostavitev podatkov prek interneta ali združitev vsebin, pridobljenih od več virov v iskalniku.

Spletne storitve SOAP so boljša rešitev za poslovne storitve, saj s specifikacijami omogočajo zanesljivo sporočanje, varnost, sinhrono in asinhrono sporočanje ter imajo podporo za transakcije. Prav tako imajo veliko povezljivost s trenutnimi poslovnimi aplikacijami in podporo standardov. Prav tako so boljša izbira, če potrebujemo operacije s stanjem.

Nobeden ni boljši, vsak je dober za svoje situacije [48]. Vprašati se moramo:

- Ali se izpostavlja podatke ali logiko?
- Ali potrebujemo WS–* razširitve?
- Kaj je boljše za razvijalce?
- Kaj je boljše dokumentirano?
- Ali imamo razvojna orodja?
- Kaj je lažje za vzdrževanje?

5 Vmesniki API za razvoj v Javi

5.1 JAX–WS

Java API for XML Web Services [49] je programski vmesnik za razvoj spletnih storitev SOAP. Uporabljen je tudi za razvoj spletnih storitev in odjemalcev, ki za komuniciranje uporabljajo XML ali RPC za izmenjavo podatkov med odjemalcem in ponudnikom storitve. Nasledil je vmesnik JAX–RPC v spletnih storitvah in aplikacijah in trenutno predstavlja jedro pri javanskem razvoju spletnih storitev. Neodvisen od operacijskega sistema in ne omejuje, odjemalec JAX–WS lahko pokliče spletno storitev, ki ne teče v Javi.

JAX–WS se nanaša na naslednje specifikacije in protokole:

- JAXB,
- SOAP 1.2,
- WSDL 1.2,
- WS–I Basic Profile,
- A metadata Facility for the Java Programming Language,
- Web Services Metadata for the Java Platform,
- Implementing Enterprise Web Services,
- Web Services Security.

JAX–WS omogoča:

- asinhrone operacije na odjemalčevi strani,
- podporo za druge načine prenašanja kot HTTP,
- poenostavitev odjemalčevega in storitvenega dostopa do sporočil izmenjave,
- podporo za upravljanje sej, ki temeljijo na sporočilih.

JAX–WS podrobno in natančno opiše, kako:

- pretvoriti WSDL 1.1 v Java metapodatke in obratno z uporabo JAXB;
- uporabiti vmesnike API samo za odjemalca, strežnik in vmesnike API, ki jih lahko uporabljata oba;
- uporabiti anotacije za označevanje razredov in metod s primeri;
- prilagoditi pretvorbe WSDL 1.1 v Java metapodatke;
- uporabiti prilagodljivo funkcionalnost za module, ki obdelujejo sporočila; v tej funkcionalnosti je vključen JAX–WS SOAP povezovanje in njegove razširitve;
- uporabiti povezave JAX–WS XML/HTTP, ki omogočajo surove možnosti za sporočanje XML prek HTTP–ja.

Referenčna implementacija pod skupnostjo GlassFish se imenuje JAX–WS RI. Je del večje zbirke, ki se uporablja za razvoj spletnih storitev SOAP in se imenuje Metro [64].

5.2 JAXP

JAXP [50] definira načine za obdelavo podatkov XML. Omogoča razčlenitev podatkov kot tok dogodkov ali tvorbo objekta, ki predstavlja podatek. Podpira XSLT, ki omogoča pretvorbo dokumenta XML v drug format in imenske prostore. Ima tri načine za razčlenitev: SAX (angl. Simple API for XML Parsing), DOM (angl. Document Object Model) in StAX (angl. Streaming API for for XML). Tabela 2 prikazuje prednosti in slabosti med njimi [51].

DOM	SAX	StAX
Pred obdelavo shrani ves dokument XML v obliki drevesa. Obdelava v katerikoli smeri.	Dogodkovno usmerjen razčlenjevalnik, ki gre od začetka in do konca po vozliščih. Ko prepozna določeno sintakso, pokliče ustrezno metodo.	Tokovno usmerjen razčlenjevalnik, ki ponuja nadzor nad začetkom, premorom in potekom razčlenjevanja. Lahko preskočimo dele dokumenta.
Zapomni si celoten dokument.	Dokumenta si ne zapomni.	Dokumenta si ne zapomni.
Porabi več spomina, slaba učinkovitost pri večjih dokumentih.	Hitrejši kot DOM, porabi manj spomina.	Hitrejši kot DOM, porabi manj spomina.
	Razčlenjevalni model PUSH	Razčlenjevalni model PULL
Branje in pisanje dokumenta	Branje dokumenta	Branje in pisanje dokumenta
Podpora CRUD		

Tabela 2: Prednosti in slabosti razčlenjevalnikov

5.3 JAXM

JAXM [52] opisuje standardni način za pošiljanje dokumentov XML prek interneta s platforme Java. Temelji na SOAP 1.1 in SAAJ ter je razširljiv za uporabo s protokoli višjih nivojev, recimo ebXML. Z uporabo JAXM odjemalec ne ve ničesar, kaj ponudnik dela v ozadju. Odjemalec kliče samo metode JAXM, za vse preostalo poskrbita ponudnik in infrastruktura. Sporočanje JAXM je lahko uresničeno tudi brez ponudnika, v tem primeru je odjemalec omejen na točka–točka pošiljanje direktno na spletno storitev. Dobil je status ostarelega (angl. »deprecated«) [53, 66].

5.4 SAAJ

SOAP with Attachments API for Java [55] je v glavnem uporabljen za sporočanje, ki se dogaja v ozadju dogodkovnih klicev implementacij JAX–WS in JAXR. Je vmesnik za razvijalce, ki želijo direktno razviti aplikacije SOAP, namesto da bi uporabili JAX–WS.

SAAJ omogoča preprosto klicanje metod in tako branje in pisanje sporočil XML, lahko tudi po internetu. Vmesnik SAAJ je skladen s SOAP 1.1 in 1.2.

5.5 JAX–RPC

Razvijalcem omogoča vključiti RPC v spletne storitve. Namen [56] je poenostaviti klicanje spletnih storitev ali aplikacij. Temelji na mehanizmu RPC. Čeprav temelji na RPC, ponuja več možnosti: lahko se pošlje cele dokumente ali samo dele dokumenta, enosmerno pošiljanje ali pošiljanje različnih sporočil. Osredotoča se na točka–točka komunikacijo SOAP. JAX–RPC je bil zamenjan z JAX–WS in ima status ostarelega (angl. »deprecated«) [57, 66].

5.6 JAXR

JAXR [58] določa enotni standard za dostop do različnih vrst registrov XML. To je infrastruktura, s katero omogočamo razvoj, uvajanje in odkritje spletnih storitev. Je nevtralen in zunanji ter ima dinamične in (poslovne) interakcije med komponentami, ki o drugih ne vedo veliko. Z JAXR razvijalci napišejo odjemalce za registre, ki so prenosni čez različne ciljne registre. Omogoča tudi vrednostno dodane zmogljivosti poleg teh, ki so v osnovnih registrih. Trenutna verzija podrobno opisuje povezavo med informacijskim modelom JAXR in obema modeloma: register ebXML in UDDI. Zaradi omejene uporabe JAXR ne upraviči status kot zahtevana komponenta. Zato je dobil status ostarelega (angl. »deprecated«) [57, 66].

5.7 JAXB

JAXB [60] predstavlja preprosti in hitri način, kako povezati sheme XML in Java predstavitve. Razvijalcem olajša vključitev podatkov XML in funkcij v Java aplikacije. Kot del tega procesa JAXB določa metode za pretvorbo dokumentov XML v javanska vsebinska drevesa in obratno. Omogoča tudi način, kako generirati shemo XML iz Java objektov.

5.8 WSIT

Skupni projekt [61] Suna in Microsofta, ki zagotavlja varnost, atomske transakcije in zanesljivo sporočanje za spletne storitve SOAP. Razširja javansko jedro za podporo XML–a in vpeljuje nove funkcije. Implementira samo del specifikacij WS–*. Zagotovi dostop do spletne storitve in njene datoteke WSDL in generiranje odjemalca iz nje. Optimizira razmerje med časom obdelave in pasovno širino pri prenosu velikih podatkov. Zagotovi, da se sporočila v danem zaporedju dostavi in da se sistem opomore v primeru izgubljenega sporočila ali prihoda sporočila v napačnem zaporedju. Implementira WS–Security, WS–SecureConversation za zagotovitev celovitosti in integritete sporočila in zaupnosti. Dodatno implementira WS–Trust in WS–Secure Policy.

5.9 Java XML Digital Signature API

Javanski standard [62] za generiranje in preverjanje veljavnosti podpisov XML. Podpis XML se lahko uporabi na podatkih vseh vrst, XML ali binarnih. Končni podpis je predstavljen v obliki XML–a. Podpis XML se uporablja za varovanje in celovitost podatkov, preverjanje pristnosti sporočila in preverjanje pristnosti podpisnika. Podpira vse potrebne in priporočljive funkcije od W3C glede XML podpisa in procesiranja. Je razširljiv in temelji na Java Cryptography Service Provider Architecture.

5.10 JAX–RS

Programski vmesnik [59] za razvoj spletnih storitev v skladu z arhitekturnim stilom REST. Uporablja anotacije za poenostavitev razvoja in uvajanje odjemalcev ter končnih točk. Z anotacijo označimo javanski razred kot sredstvo, na katerem uvedemo akcije.

Cilji vmesnika:

- Zagotovil bo nabor anotacij in razredov, ki so lahko uporabljeni s objekti POJO in so tako izraženi kot sredstva. Definira tudi življenjski cikel in obseg objekta.

- Predpostavlja, da se uporablja protokol HTTP in zagotavlja pretvorbo med elementi HTTP in URI ter ustreznimi razredi in anotacijami. Podpora za pogoste vzorce uporabe v protokolu HTTP na višjem nivoju in podpora za raznolike aplikacije HTTP–ja.

- Omogoča različne formate za tip vsebine v telesu HTTP–ja (angl. entity body content type) prek razširitev ali priključkov.

- Definira, kako so razredi razporejeni v okoljih Servlet Container in JAX–WS Provider.

- Definira okolje za razrede, ki predstavljajo sredstva in so nameščeni v Java EE Container. Opiše, kako uporabiti funkcije Java Enterprise Edition in komponente v teh razredih.

Natančno opiše:

- Kako nastaviti, potrditi in objaviti aplikacijo JAX–RS.

- Implementacije sredstev REST v javanske razrede sredstev (angl. resource classes). To so objekti POJO z uporabo anotacij. Vsak ima svoj življenjski cikel. Vsaka aplikacija JAX–RS ima eno ali več sredstev.

- Izvajalni čas JAX–RS je podaljšan z uporabo razredov ponudnika. Slednji je implementacija razširitvenega vmesnika JAX–RS. Vsaka aplikacija JAX–RS ima več ponudnikov ali nobenega.

- Zagotavlja zmogljivosti za pridobivanje in obdelavo podatkov o stanju aplikacije za uvajanje in za posamezne zahteve. Določene komponente JAX–RS morajo podpirati več hkratnih zahtev in morajo biti sposobne izbrati pravo vsebino za pravo zahtevo.

Referenčna implementacija je odprtokodni projekt pod skupnostjo GlassFish in se imenuje Jersey [65].

5.12 Primerjava JAX–WS in JAX–RS

JAX–WS [49] in JAX–RS [59] sta javanska vmesnika za razvoj spletnih storitev. JAX–WS je za razvoj spletnih storitev SOAP, JAX–RS pa za spletne storitve REST. Pri obeh uporabljamo objekte POJO in anotacije. V obeh primerih sta storitvi lahko v obliki servleta in imata končnico .war. Prav tako sta lahko kot Servlet Container ali Java EE Container.

Pri JAX–WS imamo datoteko WSDL, ki opisuje spletno storitev. Lahko jo napišemo ročno ali generiramo z različnimi orodji. Uporabljena je za pri odkrivanju in klicanju spletne storitve. Lahko celo generiramo odjemalca iz te datoteke. WADL pa še ni vključen v JAX–RS 1.1.

JAX–WS se nanaša na JAXB, SOAP 1.2, WSDL 2.0, WS–I Basic Profile, A Metadata Facility for the Java Programming Language, Implementing Enterprise Web Services, Web Services Security. JAX–RS se ne nanaša na nobeno specifikacijo, lahko jih pa uporablja, recimo JAXB.

Sporočanje pri JAX–WS je lahko asinhrono in sinhrono, kar omogoča protokol SOAP. Podpira asinhrone operacije za odjemalca. Protokol HTTP, ki ga uporablja JAX–RS, pa omogoča samo sinhrono sporočanje.

JAX–WS ima podporo za razvoj storitve in odjemalca, kjer ima JAX–RS podporo samo za razvoj storitve.

Oba imata abstraktni razred, ki zagotovi različne metode za kreiranje objektov kot končno točko storitve. V JAX–RS se imenuje RuntimeDelegate, v JAX–WS pa ServiceDelegate. Uporabljata se kot alternativa servletu.

JAX–WS omogoča dve izbiri za pošiljanje sporočil. Pošilja se lahko prek SOAP ali sporočila XML prek protokola HTTP. JAX–RS je omejen na HTTP in sporočila XML.

JAX–RS podpira številne vhodne in izhodne vrste medijev [67], na primer XML, JSON, plain text, HTML, atom in druge. JAX–WS uporablja samo XML.

Tabelarični prikaz opisanih podobnosti in razlik

JAX–WS	JAX–RS
Razvoj spletnih storitev SOAP	Razvoj spletnih storitev REST
Uporablja objekte POJO in anotacije.	Uporablja objekte POJO in anotacije.
V obliki servleta, servlet containerja ali Java EE Container.	V obliki servleta, servlet containerja ali Java EE Container.
Opisna datoteka WSDL, iz katere lahko generiramo odjemalca.	Opisna datoteka WADL, ki še ni vključena v JAX–RS 1.1.
Nanaša se na JAXB, SOAP 1.2, WSDL 2.0, WS–I Basic Profile, A Metadata Facility for the Java Programming Language, Implementing Enterprise Web Services, Web Services Security.	Ne nanaša se na nobeno specifikacijo, lahko jih pa uporablja, recimo JAXB.
Sporočanje je asinhrono ali sinhrono	Samo sinhrono sporočanje
Podpora za razvoj storitve in odjemalca	Podpora za razvoj storitve
Abstraktni razred ServiceDelegate za kreiranje objektov kot končno točko storitve	Abstraktni razred RuntimeDelegate za kreiranje objektov kot končno točko storitve
Sporočila se lahko pošilja prek SOAP ali sporočila XML prek HTTP.	Omejen na sporočila XML in HTTP.
Podpora samo formatu XML	Podpira XML, JSON, plain text, HTML, atom …

Tabela 3: Podobnosti in razlike med JAX–WS in JAX–RS

5.12.1 Anotacije

Oba uporabljata anotacije, s katerimi poimenujemo razrede in metode. JAX–RS priskrbi anotacije za poimenovanje poti za URI, določitev uporabe metod ob klicih ukazov HTTP, pridobivanje vrednosti iz parametrov, glave in vsebine in drugo. Anotacije JAX–WS določajo zajemanje komponente za ovojnice, povezovanje s specifično spletno storitvijo, kontrolo

uporabe nekaterih pomožnih protokolov, določevanje razreda kot spletno storitev ali ponudnik, določevanje zajemanja komponent za ovojnice zahtev in drugo.

JAX–RS vsebuje anotacije za poimenovanje.

Anotacija REST	**Opis**
GET	Določa metodo ob klicu ukaza GET.
POST	Določa metodo ob klicu ukaza POST.
PUT	Določa metodo ob klicu ukaza PUT.
DELETE	Določa metodo ob klicu ukaza DELETE.
HEAD	Določa metodo ob klicu ukaza HEAD.
Consumes	Določa seznam vhodnih vrst MIME.
Produces	Določa seznam izhodnih vrst MIME.
ApplicationPath	Določa pot, ki tvori osnovni URI za vse korenske razrede.
Path	Relativna pot za sredstvo. Uporabljen na razredu ustvari korensko sredstvo, uporabljen na metodi identificira pod-sredstvo.
PathParam	Omogoča pridobivanje vrednosti s poti URI.
QueryParam	Omogoča pridobivanje vrednosti s parametra query v URI–ju.
FormParam	Omogoča pridobivanje vrednosti s parametra form v telesu zahteve.
MatrixParam	Omogoča pridobivanje vrednosti s parametra matrix v URI–ju,
CookieParam	Omogoča pridobivanje vrednosti s piškotka HTTP–ja.
HeaderParam	Omogoča pridobivanje vrednosti z glave HTTP–ja.
Encoded	Onemogoči avtomatično kodiranje za pot in parametre query, form in matrix.
DefaultValue	Določa privzeto vrednost za polje, lastnost ali metodo.
Context	Pridobivanje vrednosti z URI–ja, glave, zahteve in varnostne vsebine.
HttpMethod	Določa metode HTTP za zahteve.
Provider	Določa, da označeni razred implementira razširitev interface.

Tabela 4: Anotacije JAX–RS

JAX–WS vsebuje anotacije za poimenovanje.

Anotacija SOAP	**Opis**
ServiceMode	Definiranje načina za razrede ponudnika.
WebFault	Določa povezovanje napak WSDL z izjemami Java.
RequestWrapper	Določa zajemanje komponente za ovojnice zahtev, ustvarjene z JAXB, in elementa ime ter imenskega prostora za transformacijo med spominsko predstavitvijo in formatom, primernim za aplikacijo in obratno komponente.
ResponseWrapper	Določa zajemanje komponente za ovojnice odgovorov, ustvarjene z JAXB, in elementa ime ter imenskega prostora za transformacijo med spominsko predstavitvijo in formatom, primernim za aplikacijo in obratno komponente.
WebServiceClient	Določa povezovanje razreda s specifično spletno storitvijo, identificirano z URL do dokumenta WSDL in kvalificiranim imenom.
WebEndpoint	Določa povezovanje metod razreda s specifičnim wsdl:port, ki ima svoje ime.
WebServiceProvider	Določa označevanje razreda, da je ponudnik.
BindingType	Določevanje uporabe povezovanja razredov končnih točk.
WebServiceRef	Določa sklicevanje na spletno storitev.
WebServiceRefs	Določitev več sklicevanj na spletne storitve na enem razredu.
WebService	Označi razred kot implementacijo spletne storitve ali označi vmesnik Java kot definiranje vmesnika spletne storitve.
WebMethod	Določi metodo, da je operacija spletne storitve.
Oneway	Označuje, da ima metoda samo vhodno sporočilo.

WebParam	Prilagodi povezovanje med parametrom z elementi XML ter s sporočilnim delom.
WebResult	Prilagodi povezovanje vrnjene vrednosti z delom WSDL in elementom XML.
SOAPBinding	Določa povezovanje spletne storitve na sporočilni protokol SOAP.
HandlerChain	Poveže spletno storitev z zunanjim krmilnikom.
Action	Določevanje WS–Addressing vrednosti vhodom, izhodom in napakam, povezanih z metodo.
FaultAction	Določevanje akcije WS–Addressing za storitveno specifično izjemo.
WebServiceFeatureAnnotation	Določa identificiranje drugih anotacij kot funkcije spletnih storitev. Je metaanotacija.
Addressing	Določa kontroliranje uporabe WS–Addressing.
MTOM	Določa kontroliranje uporabe MTOM.
RespectBinding	Določa kontroliranje, karkoli se mora upoštevati, ki je v wsdl:binding povezano s končno točko.

Tabela 5: Anotacije JAX–WS

5.12.2 Razširitve

Obe vrsti vmesnika uporabljata razširitve. JAX–RS to stori prek ponudnikov (angl. provider), JAX–WS pa s pomočjo krmilnikov (angl. handler). Krmilniki so prestrezniki sporočil za dodatno obdelavo vhodnih in odhodnih sporočil. Ponudniki prestrezajo vezavo med predstavitvami, prestrezajo izjeme ali zagotovijo določeno vsebino.

JAX–RS ima tri vrste ponudnikov:

- Ponudnik entitet

  Ponudnik entitet povezuje med predstavitvami sredstev in njihovimi povezanimi javanskimi tipi. Poznamo bralca (angl. MessageBodyReader) in zapisovalca (angl. MessageBodyWritter). Bralec je pogodba med JAX–RS v času izvajanja in komponentami, ki zagotovijo povezavo med predstavitvami in Java tipi. Zapisovalec je pogodba med JAX–RS v času izvajanja in komponentami, ki zagotovijo povezavo med javanskimi tipi in predstavitvami.

- Ponudnik vsebine
  Dobavlja vsebino razredom sredstev in drugim ponudnikom.

- Ponudnik za izjeme
  Omogoča prilagajanje ujemanja izjem. Ob izjemi se poveže z instanco Response in vrne njen odgovor.

JAX–WS ima dve vrsti krmilnikov:

- Logični
  Operirajo samo na lastnostih sporočilne vsebine in sporočila PAYLOAD. Ne morejo spremeniti delov, specifičnih za protokol.

- Protokolni
  Operirajo samo na lastnostih sporočilne vsebine in sporočilih, specifičnih za protokol. Dostopajo in spreminjajo lahko samo vidike, specifične za protokol.

Krmilnik ima dve razširitvi:

- Vez s SOAP
  Krmilnik povežemo s SOAP lahko na dva načina. Prvi način je programski in poteka tako, da uporabimo vmesnike, ki jih ponuja JAX–WS za nastavitev odjemalca. Strežnik se nastavi z anotacijami. Lahko uporabimo logični krmilnik ali krmilnik SOAP. Drugi način je z uporabo modela obdelave. To je arhitektura spletnih storitev za Java Enterprise Edition.

- Vez z XML/HTTP
  Namesto SOAP se lahko uporablja tudi XML/HTTP. Programski način povezovanja je samo za odjemalce in poteka prek vmesnikov, ki jih ponuja JAX–WS. Uporabi se lahko samo logični krmilnik. Strežnik se nastavi z uporabo modela obdelave.

5.12.3 Glave

JAX–WS lahko dostopa in manipulira z glavami SOAP in HTTP, pri čemer JAX–RS dostopa in manipulira samo s polji glav HTTP v omejenemu obsegu.

JAX–RS podpira naslednja polja glave HTTP:

– Accept,
– Accept–Charset,
– Accept–Encoding,
– Accept–Language,
– Allow,
– Authorization,

– Cache–Control,
– Content–Encoding,
– Content–Language,
– Content–Length,
– Content–Type,
– Cookie,
– Date,
– ETag,
– Expires,
– Expect,
– If–Match,
– If–Modified–Since,
– If–None–Match,
– If–Unmodified–Since,
– Last–Modified,
– Location,
– Set–Cookie,
– Transfer–Encoding,
– Vary,
– WWW–Authenticate.

6 Praktični primer

6.1 Orodja in vmesniki

6.1.1 NetBeans

Netbeans [68] je odprtokodno integrirano razvojno okolje za razvoj aplikacij v javanskem okolju in C/C++. Na voljo je za Windows, Mac, Linux in Solaris. Dobra podpora za razvoj poslovnih in mobilnih aplikacij, prav tako podpira PHP, JavaScript, Ajay, Groovy in Grails. Z razširitvami in s knjižnicami pa še dodatno povečamo podporo. Druge funkcije vključujejo graditelj GUI in podporo CVS. Je zelo dobro dokumentirano in ima podporo skupnosti. Verzija 7.0.1 je prinesla izboljšano integracijo z Oracle Web Logic server in GlassFish 3.1 ter novosti Maven 3 in podporo za urejanje HTML5.

6.1.2 GlassFish

Je [69] odprtokodni aplikacijski strežnik, trenutna verzija je 3.1.1. Projekt je začel Sun Microsystems za platformo Java EE, zdaj ga sponzorira Oracle. Podpira vse potrebne vmesnike Java Enterprise Edition, kot so na primer JDBC, RMI in e–Mail, ter definira, kako jih povezati in koordinirati za delovanje. Omogoča tudi druge funkcije, specifične za Java Enterprise Edition komponente: JavaBean, Connector, servlet, portlest in Page. Omogoča razvoj poslovnih aplikacij, ki so prenosljive in razširljive ter se jih dá vključiti v druge tehnologije.

6.1.3 WampServer

Je [70] skupek treh tehnologij: strežnika Apache, PHP-ja in podatkovne baze MySQL. Z njim si postavimo podatkovno bazo. Vključuje tudi PHPMyAdmin za upravljanje podatkovne baze. Vse tri tehnologije v paketu so odprtokodne.

6.1.4 Jersey

Jersey [65] je odprtokodna referenčna implementacija za razvoj spletnih storitev REST. Ponudi tudi vmesnik, s katerim lahka razširimo funkcionalnost za svoje potrebe. Implementira podporo za uporabo anotacij, ki olajša delo razvijalcem v okolju Java.

6.1.5 Metro

Metro [64] je visoko zmogljiva in razširljiva zbirka komponent in vmesnikov. Vključuje JAXB RI, JAX–WS RI, SAAJ RI, SJSXP in WSIT ter vse potrebne knjižnice.

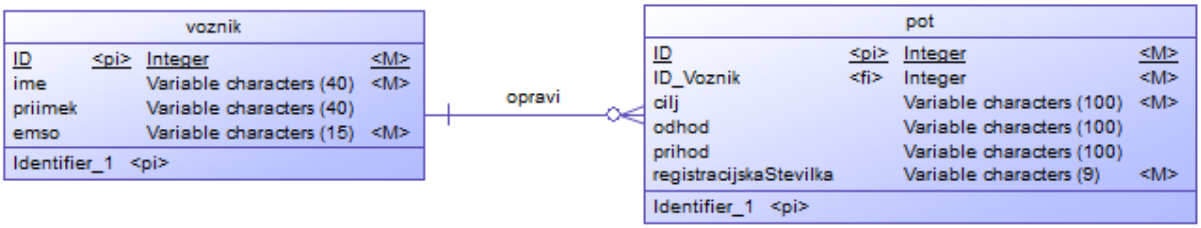
6.1.6 Firefox in razširitev RESTClient

Priročna razširitev [71], ki omogoča interakcijo s spletnimi storitvami REST in z njenimi sredstvi. Omogoča pošiljanje zahtev in tudi nastavitev vsebine za pošiljanje, glave in parametrov.

6.2 Opis primera

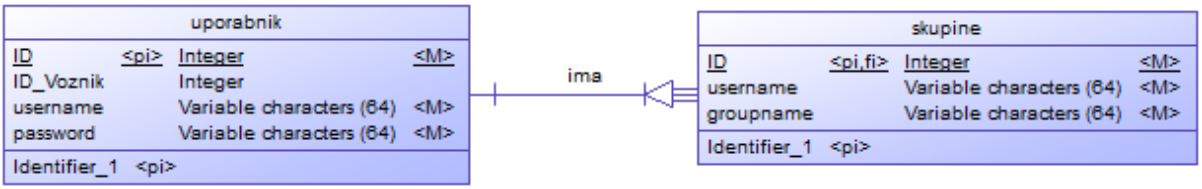
Za primer spletne storitve smo vzeli vožnje, torej voznike in poti, ki jih prevozijo. Vsak voznik in vsaka pot sta zapisani v podatkovni bazi v dveh tabelah, kot je razvidno iz slike 6.

Prva ima naziv voznik in opisuje voznike z imenom, s priimkom in matično številko. Druga pa ima podatke o poteh za voznike. Pot ima atribute cilj, odhod, prihod in registracijska številka vozila.



Slika 6: Tabeli voznik in pot

Vsak od voznikov ima tudi svoje uporabniško ime in geslo za prijavo v spletno storitev, kot prikazuje slika 7. Uporabniško ime je povezano z imenom skupine, ki omejuje oziroma dovoljuje dostop. Vozniki sodijo pod vlogo USERS. Drugi uporabniki, recimo administratorji ali nadzorniki, imajo vlogo ADMINISTRATORS. Vozniki lahko samo gledajo svoje zapise v bazi, drugi pa lahko vanjo pišejo, brišejo ali popravljajo zapise in gledajo vse zapise.



Slika 7: Tabeli uporabnik in skupine

6.3 Primer SOAP

6.3.1. Storitev

6.3.1.1 Razvoj storitve

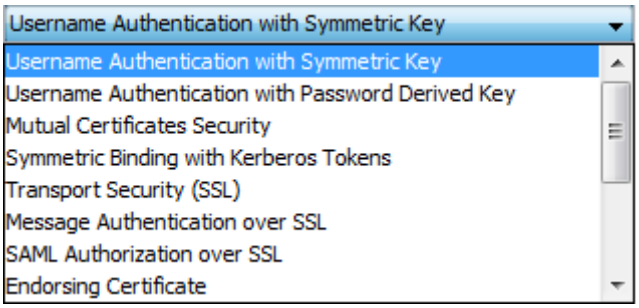
Z uporabo vmesnika Metro je spletna storitev realizirana kot javanski razred, pri tem je pomembno, da ima anotacijo WebService, s čimer se označi razred kot implementacijo spletne storitve. V tem razredu se kreira metode. Slednje morajo imeti anotacijo WebMethod, ki označi, da je del te spletne storitve. Če metoda potrebuje parametre, to storimo z anotacijo WebParam in jih ustrezno poimenujemo. Slika 8 prikazuje metode in pripadajoče parametre spletne storitve.

```
    @WebMethod(operationName = "createVoznik")
    public String createVoznik(@WebParam(name = "ime") String ime,
                               @WebParam(name = "priimek") String priimek,
                               @WebParam(name = "emso") String emso) {...}

    @WebMethod(operationName = "editVoznik")
    public String editVoznik(@WebParam(name = "id") Integer id,
                             @WebParam(name = "ime") String ime,
                             @WebParam(name = "priimek") String priimek,
                             @WebParam(name = "emso") String emso) {...}

    @WebMethod(operationName = "removeVoznik")
    public String removeVoznik(@WebParam(name = "id") Integer id) {...}

    @WebMethod(operationName = "najdiVoznika")
    public String najdiVoznika(@WebParam(name = "id") Integer id) {...}

    @WebMethod(operationName = "najdiVseVoznike")
    public String najdiVseVoznike() {...}

    @WebMethod(operationName = "createPot")
    public String createPot(@WebParam(name = "idV") Integer idV,
                            @WebParam(name = "cilj") String cilj,
                            @WebParam(name = "odhod") String odhod,
                            @WebParam(name = "prihod") String prihod,
                            @WebParam(name = "regSt") String regSt) {...}

    @WebMethod(operationName = "editPot")
    public String editPot(@WebParam(name = "idP") Integer idP,
                          @WebParam(name = "idV") Integer idV,
                          @WebParam(name = "cilj") String cilj,
                          @WebParam(name = "odhod") String odhod,
                          @WebParam(name = "prihod") String prihod,
                          @WebParam(name = "regSt") String regSt) {...}

    @WebMethod(operationName = "removePot")
    public String removePot(@WebParam(name = "id") Integer id) {...}

    @WebMethod(operationName = "najdiPot")
    public String najdiPot(@WebParam(name = "id") Integer id) {...}

    @WebMethod(operationName = "findPotByVoznikID")
    public String findPotByVoznikID(@WebParam(name = "user") String user) {...}

    @WebMethod(operationName = "findPotByVoznikIDinID")
    public String findPotByVoznikIDinID(@WebParam(name = "user") String user,
                                        @WebParam(name = "id") Integer id) {...}

    @WebMethod(operationName = "najdiVsePoti")
    public String najdiVsePoti() {...}
}
```

Slika 8: Metode spletne storitve

6.3.1.2 Varnost

Za varnost poskrbimo z WSIT, ki je del vmesnika Metro. WSIT priskrbi kar nekaj možnosti za varnost. Slika 9 jih kaže samo nekaj.



Slika 9: Varnostne možnosti

Izbrali smo si prvo možnost, avtorizacijo z uporabniškim imenom s simetričnim ključem (angl. Username Authentication with Symmetric key). Kriptografija simetričnega ključa temelji na skupnem tajnem ključu, ki se uporablja za podpis in šifriranje sporočila. Zaščiti integriteto in zaupnost sporočila.

6.3.1.3 Testiranje in gledanje sporočil

NetBeans in GlassFish omogočata testiranje spletnih storitev, preden se doda varnost z WSIT. Strežnik GlassFish ima za vsako spletno storitev svojo testno stran, ki jo odpre v privzetem brskalniku. Do nje dostopamo prek bližnjice v NetBeansu ali v administrativni konzoli strežnika GlassFish. Slika 10 prikazuje zahtevo in odgovor SOAP brez varnosti, potem ko smo poklicali funkcijo najdiVoznika s parametrom 5.

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:najdiVoznika xmlns:ns2="http://Storitev/">
            <id>5</id>
        </ns2:najdiVoznika>
    </S:Body>
</S:Envelope>
```

SOAP Response

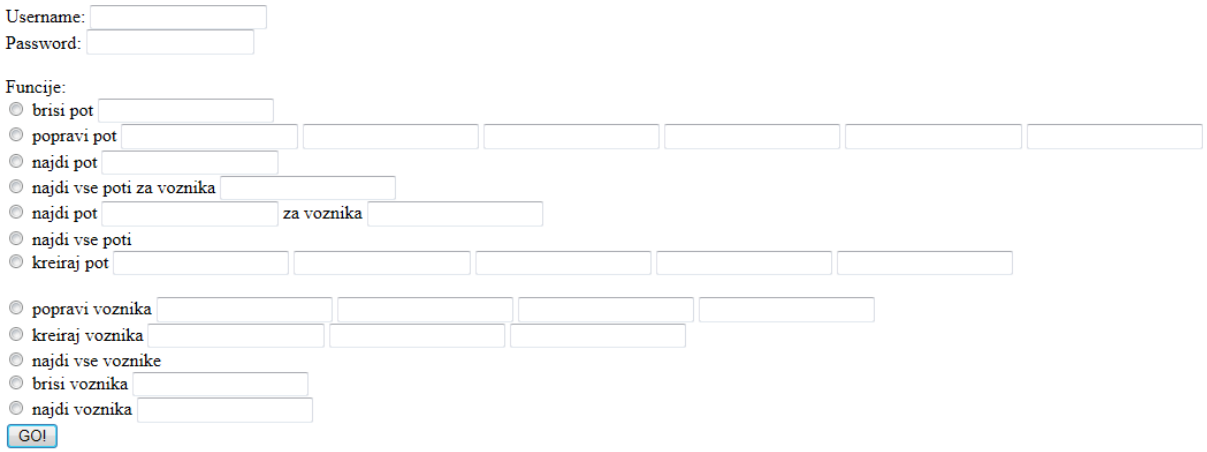
```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:najdiVoznikaResponse xmlns:ns2="http://Storitev/">
            <return><?xml version="1.0" encoding="UTF-8" standalone="yes"?><Voznik><emso>061095050033</emso><id>5</id><ime>Miha</ime><priimek>Kosir</priimek></Voznik></return>
        </ns2:najdiVoznikaResponse>
    </S:Body>
</S:Envelope>
```

Slika 10: Zahteva in odgovor SOAP

Po dodani varnostni možnosti WSIT tako testiranje ni več mogoče. Obstaja možnost nastavitve strežnika GlassFish, tako da izpiše zahtevo in odgovor v konzolo NetBeansa. Zahteva in odgovor SOAP imata po dodani varnosti v glavi podatke o ključu in algoritmih za (de)šifriranje. Zahteva ima v telesu šifrirane podatke za iskanje, odgovor pa šifrirane podatke, po katerih poizvedujemo.

6.3.2. Razvoj odjemalca

Razvoj odjemalca se začne, ko imamo lokacijo datoteke WSDL. Lokacija za našo spletno storitev je http://localhost:8080/SpletnaStoritev/SpletnaStoritev?wsdl. V odjemalcu Metro se na spletno storitev sklicujemo z anotacijo WebServiceRef, ki za parameter vzame lokacijo URL za datoteko WSDL. Na tak način se ustvari objekt, ki predstavlja storitev in njene metode. Potem smo metode poklicali v servletu. Ta služi kot razširitev zmogljivosti strežnika, ki gosti dostop do aplikacij prek modela zahteva–odgovor. Uporabniški vmesnik je stran JSP in omogoča vnos podatkov in izbiro za klicanje metod. Kot prikazuje slika 11, na tej strani vnesemo uporabniško ime in geslo, izberemo funkcijo ter ji vpišemo parametre, če jih ima.



Slika 11: Stran JSP odjemalca SOAP

6.4 Primer REST

6.4.1. Storitev

6.4.1.1 Razvoj storitve

Razvoj spletne storitve REST poteka v treh korakih:

1) ugotovitev nabora podatkov in razdelitev v sredstva;
2) vsako sredstvo se:
    a. poimenuje z URI;
    b. izpostavi podmnožico enotnega vmesnika (izberemo enotne operacije);
    c. določi, kateri podatki in v kakšni obliki bodo poslani do odjemalca ali sprejeti od odjemalca.
3) razmisliti potek dogodkov: kaj naj bi se zgodilo in kaj lahko gre narobe.

Korak 1

Naš primer storitve REST za nabor podatkov vzame podatkovno bazo s tabelama voznik in pot s podatki, ki jih bomo izpostavljali. Potem lahko definiramo dve sredstvi, ki sta obe tabeli: voznik in pot.

Korak 2

Sredstva so imenovana z URI, v našem primeru je to URL. Naš korenski URI za dostop do spletne storitve je http://localhost:8080/RestSpletnaStoritev/voznje/. Podatki iz obeh tabel so

predstavljeni kot predstavitve sredstev in so v formatu XML ali JSON. Predstavitve sredstev dobimo tako, da pokličemo imena sredstev z ukazi HTTP.

Sredstvo voznik je poimenovano z naslednjimi URI–ji in s pripadajočimi ukazi HTTP.

Relativni URI	ukaz HTTP
/vozniki/	GET, POST, PUT
/vozniki/{user}	DELETE, GET

Tabela 6: Prikaz URI–jev in pripadajočih ukazov za voznika

Sredstvo pot je poimenovano z naslednjimi URI–ji in s pripadajočimi ukazi HTTP.

Relativni URI	ukaz HTTP
/pot	GET, POST, PUT
/pot/{id}	DELETE, GET
/voznik/{user}/pot	GET
/ voznik/{user}/pot/{id}/	GET

Tabela 7: Prikaz URI–jev in pripadajočih ukazov za pot

Z uporabo vmesnika Jersey je vsako sredstvo realizirano kot javanski razred, pri tem je pomembno, da ima anotacijo Path, s čimer zaznamuje ta razred kot korensko sredstvo. V tem razredu se potem kreira metode. Če metoda nima anotacije Path, je potem dosegljiva na URI–ju razreda. Če pa jo ima, pa identificira pod-sredstvo in ima svojo relativno pot.

Metoda mora imeti tudi eno od štirih anotacij GET, PUT, POST, DELETE, ki določa, na kateri klic bo odgovorila.

Prav tako lahko metodi določimo tip sporočila, ki ga prejme kot vhod, in tip sporočila, ki ga vrne. To storimo z anotacijama Consumes in Produces.

Slika 12 prikazuje primer metode, ki najde določeno pot za določenega voznika. Če pogledamo anotacije, metoda odgovori na ukaz GET in poimenuje svojo pod-sredstvo. Sprejme zahtevo in vrne odgovor v formatu XML ali JSON. Parametre sprejme z URI–ja, to sta parametra user in id.

```
@GET
@Path("/voznik/{user}/pot/{id}")
@Consumes({"application/xml", "application/json"})
@Produces({"application/xml", "application/json"})
public Response findPotByVoznikIDinID(@PathParam("user") Integer user, @PathParam("id") Integer id) {

    if( security.isUserInRole("ADMINISTRATORS") ) {

        Query q = getEntityManager().createQuery("SELECT p FROM Pot p WHERE p.iDVoznik = :user AND p.id = :id");
        q.setParameter("user", new Voznik(user));
        q.setParameter("id", id);
        List<Pot> list = q.getResultList();
        if(!list.isEmpty())
            return Response.status(Response.Status.OK).entity(list.toArray(new Pot[list.size()])).build();
        else
            return Response.status(Response.Status.NOT_FOUND).build();
    }
    return Response.status(Response.Status.UNAUTHORIZED).build();
}
```

Slika 12: Metoda za iskanje poti

Korak 3

Potrebno je razmisliti o poteku dogodkov, še posebej tudi o napakah in izjemah, do katerih lahko pride in jih primerno ujeti ali izpisati. Odjemalec lahko poda slabo formatirane in nesmiselne podatke ali jih sploh ne pošlje, lahko pa pošlje tudi napačne podatke za avtorizacijo. Lahko pride tudi do napake na strežniku. Nekatere pokrije že Java, Jersey ali strežnik, preostale moramo sami.

Razvidno pri metodi, ki jo prikazuje slika 12, smo ujeli izjeme in vrnili opozorila, če ni iskanega zapisa ali če uporabnik ni avtoriziran.

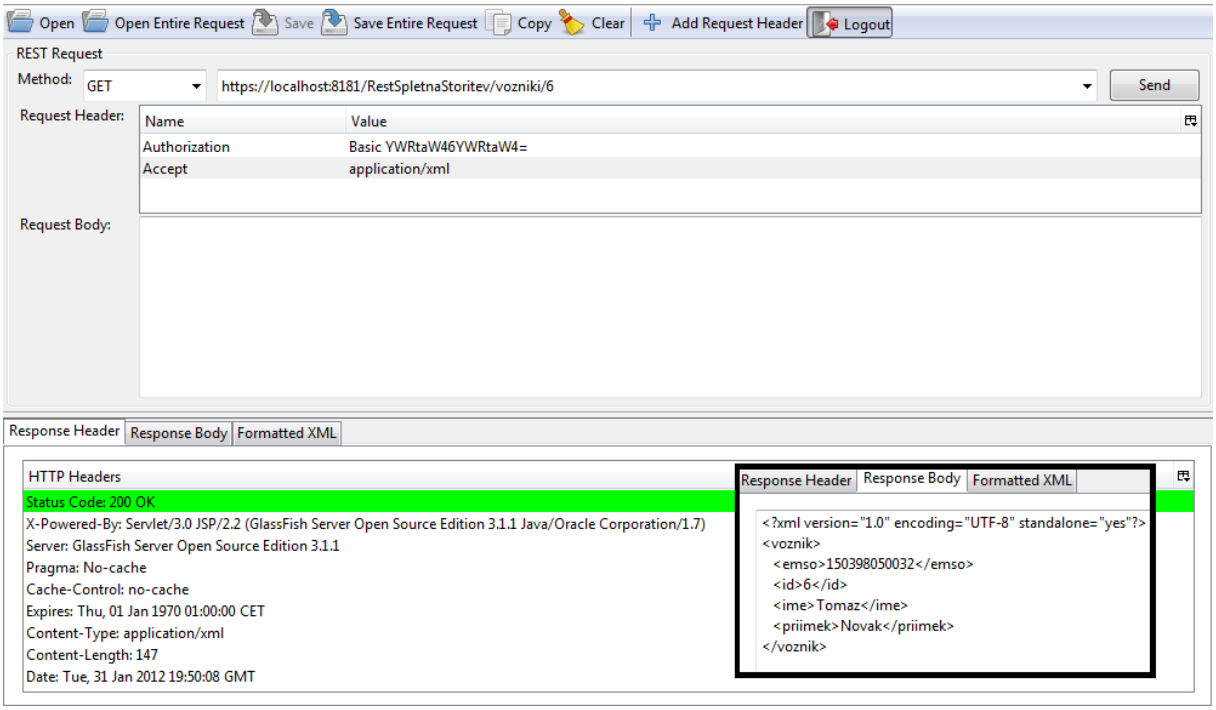
6.4.1.2 Varnost

Naš primer vključuje osnovno avtorizacijo HTTP. To je metoda, ki zagotovi uporabniško ime in geslo pri zahtevi. Preveri se na prejemnikovi strani. Če je avtorizacija neuspešna, se dostop zavrne, drugače se odobri. Za potek avtorizacije poskrbi strežnik GlassFish. Za delovanje mu moramo definirati dovoljene uporabnike prek postavitve JDBCRealm. Kako se ga nastavi, je obrazloženo v prilogi 1. Prav tako smo uporabili SSL, ki zagotovi varno komunikacijo. Naš korenski URI se spremeni in postane https://localhost:8181/RestSpletnaStoritev/voznje/.

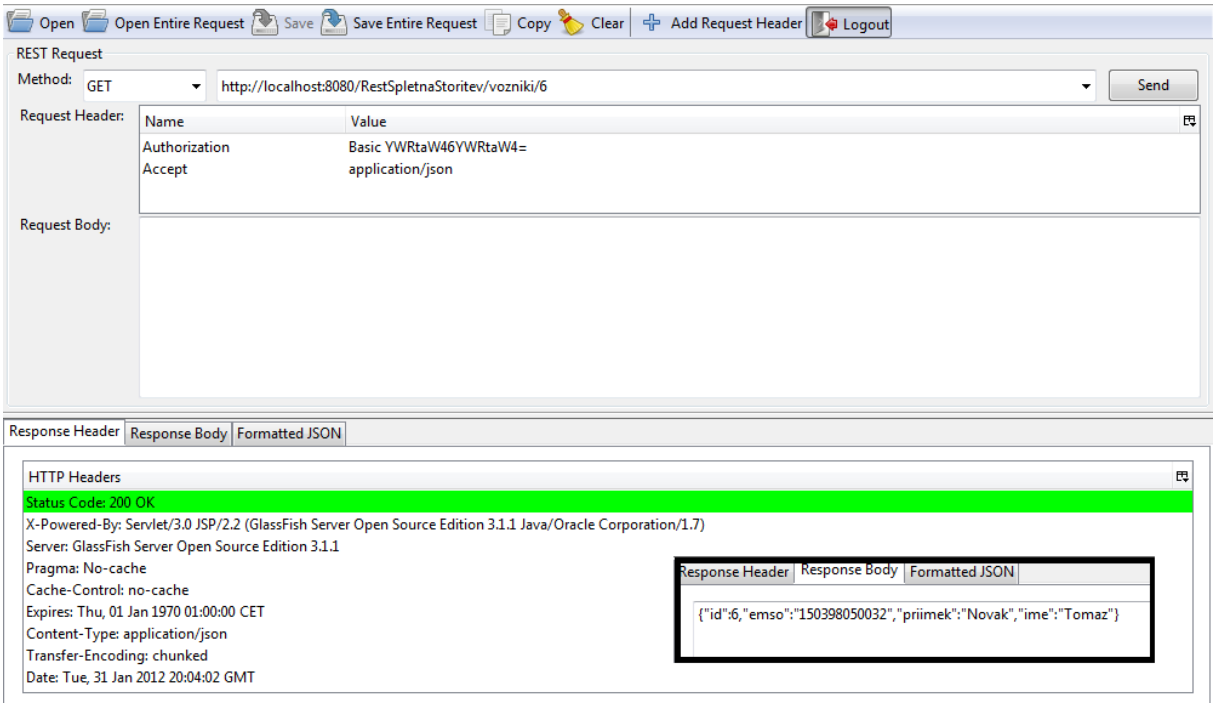
6.4.1.2 Testiranje in gledanje sporočil

Za to dejanje sem uporabil Firefox in razširitev RESTClient, omogočata pa ga tudi NetBeans in GlassFish, vendar je omejeno. Realizirano je v brskalniku, ki ne dovoli pošiljanja zahteve DELETE in PUT. Z razširitvijo RESTClient smo lahko pošiljali vse zahteve na spletno storitev, testirali in opazovali odgovore.

Slika 13 in 14 prikazujeta pošiljanje zahteve in prejeti odgovor na to zahtevo. Poslali smo zahtevo GET na URI https://localhost:818/RestSpletnaStoritev/vozniki/6. V glavo zahteve smo dodali dva polja. Polje Authorization določa uporabniško ime in geslo za avtorizacijo HTTP. Polje Accept pove strežniku, da hočemo odgovor v določenemu formatu. Na sliki 13 je prikazan odgovor v formatu XML, na sliki 14 pa odgovor v formatu JSON.



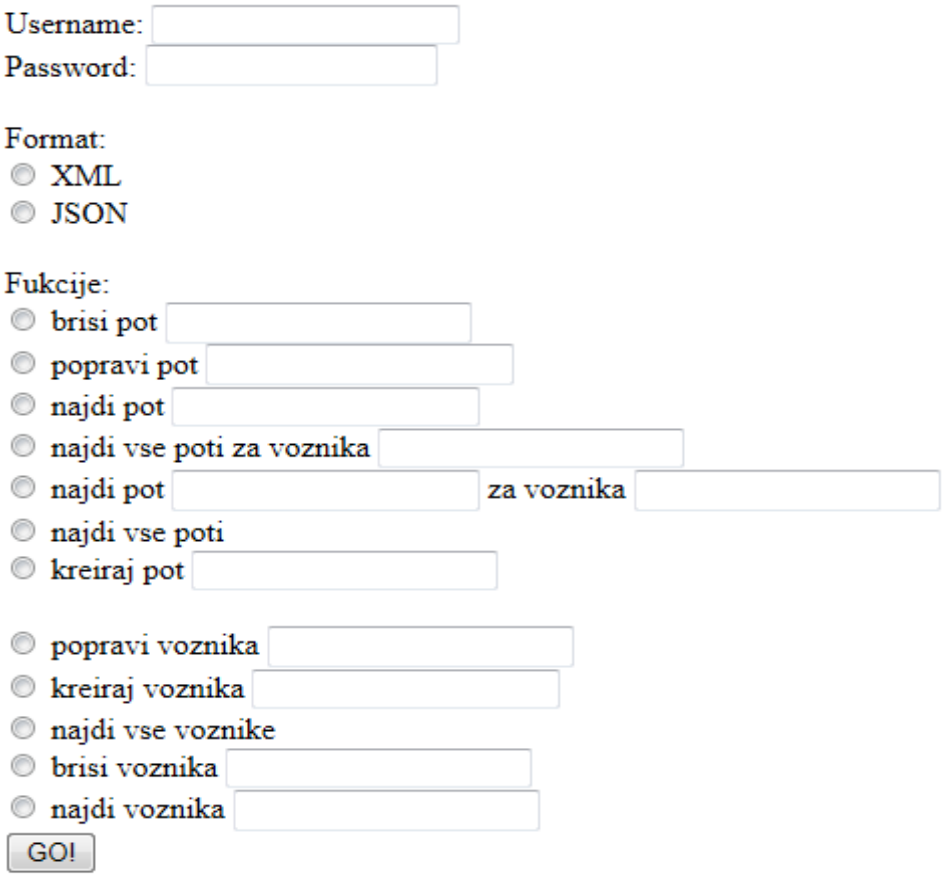
Slika 13: Pošiljanje zahteve GET in odgovor XML



Slika 14: Pošiljanje zahteve GET in odgovor JSON

6.4.2 Razvoj odjemalca

Odjemalca smo razvili z vmesnikom Jersey. Ta nam ustvari razred za vsako sredstvo z vsemi metodami, ki opravijo klice HTTP. Metode smo poklicali v servletu. Slednji služi kot razširitev zmogljivosti strežnika, ki gosti dostop do aplikacij prek modela zahteva–odgovor. Uporabniški vmesnik je stran JSP, ki jo prikazuje slika 15. Omogoča vnos podatkov in izbiro za klicanje. Na tej strani vnesemo uporabniško ime in geslo, izberemo format odgovora, ki ga želimo, in izberemo funkcijo ter ji vpišemo parametre, če jih ima.



Slika 15: Stran JSP odjemalca REST

Slika 16 prikazuje dve metodi odjemalca, ki nam ju generira Jersey. Obe metodi opravita ukaz GET na https://localhost:8181/RestSpletnaStoritev/vozniki/{user}, le da prva vrne odgovor v obliki XML, druga pa v obliki JSON.

```
public <T> T find_XML(Class<T> responseType, String user) throws UniformInterfaceException {
    WebResource resource = webResource;
    resource = resource.path(java.text.MessageFormat.format("{0}", new Object[]{user}));
    return resource.accept(javax.ws.rs.core.MediaType.APPLICATION_XML).get(responseType);
}

public <T> T find_JSON(Class<T> responseType, String user) throws UniformInterfaceException {
    WebResource resource = webResource;
    resource = resource.path(java.text.MessageFormat.format("{0}", new Object[]{user}));
    return resource.accept(javax.ws.rs.core.MediaType.APPLICATION_JSON).get(responseType);
}
```

Slika 16: Metodi odjemalca za iskanje voznika

7. Zaključek

Namen diplomske naloge je bil opis razvoja spletnih storitev. Opisali smo spletni storitvi REST in SOAP. Pri storitvah SOAP smo se osredotočili na protokol SOAP, opisni jezik WSDL, specifikacije WS–* in WS–I Basic Profile. Pri storitvah REST pa smo opisali arhitekturni stil REST in format JSON. Izpostavili smo tudi njune prednosti in pomanjkljivosti ter jih primerjali. Za obe vrsti spletnih storitev smo pregledali tudi vmesnike API, ki nam poenostavijo razvoj. Omejili smo se na programski jezik Java.

Razvili smo tudi spletno storitev REST, ki je za sredstva vzela podatke iz podatkovne baze. Te sredstva smo poimenovali z URI–jem, na njih izvajali operacije HTTP ter z njimi manipulirali. Seveda so to lahko delali samo pooblaščeni uporabniki. Za varnost smo uporabili osnovno avtorizacijo HTTP ter omejili dostop do pisalnih funkcij samo za administratorje. Za varno pošiljanje sporočil smo uporabili protokol SSL. Razvili smo tudi odjemalca, ki opravi klice HTTP in izpiše odgovore. Pri razvoju nismo naleteli na težave ali dodatne izzive.

Prav tako smo razvili spletno storitev SOAP, ki vsebuje metode za pregledovanje in spreminjanje zapisov v tabelah v podatkovni bazi. Za varnost smo uporabili vmesnik WSIT in njegovo možnost avtorizacije z uporabniškim imenom s simetričnim ključem. Generirali smo tudi odjemalca za našo storitev iz datoteke WSDL. Pri razvoju nismo imeli težav. Če bi uporabili kakšno drugo varnostno opcijo, bi imeli več dela s certifikati in z nastavitvami.

Danes so spletne aplikacije razširjene na spletu in v poslovnemu svetu. Zaradi svoje neodvisnosti, uporabe standardnih in tekstovnih protokolov in splošne povezljivosti z drugimi storitvami ustvarijo hiter tok informacij za izmenjavo. Če dodamo še nižje cene razvoja in vzdrževanja, zahvaljujoč spletu in uporabi standardnih protokolov, je to dobra poslovna rešitev za podjetja. Med najbolj znanimi na spletu so prav gotovo Google Maps, Amazon Simple Storage Device, uporabljajo pa jih tudi Yahoo, eBay in fedEx.

Priloge

Priloga 1: Nastavitve strežnika GlassFish za avtorizacijo

Priloga 1: Nastavitve strežnika GlassFish za avtorizacijo

Realm Name: userautho

Class Name: com.sun.enterprise.security.auth.realm.jdbc.JDBCRealm

Properties specific to this Class

JAAS Context: *	jdbcRealm Identifier for the login module to use for this realm
JNDI: *	VoznjeBaza JNDI name for this realm
User Table: *	uporabnik Name of the database table that contains the list of authorized users for this realm
User Name Column: *	username Name of the column in the user table that contains the list of user names
Password Column: *	password Name of the column in the user table that contains the user passwords
Group Table: *	skupine Name of the database table that contains the list of groups for this realm
Group Name Column: *	groupname Name of the column in the group table that contains the list of group names
Assign Groups:	Comma-separated list of group names
Database User:	Specify the database user name in the realm instead of the JDBC connection pool
Database Password:	Specify the database password in the realm instead of the JDBC connection pool
Digest Algorithm:	MD5 Digest algorithm (default is SHA-256); note that the default was MD5 in GlassFish versions prior to 3.1
Encoding:	Encoding (allowed values are Hex and Base64)
Charset:	UTF-8 Character set for the digest algorithm

Viri

[1] Uvod v spletne storitve. Dostopno na spletnem naslovu: http://docs.oracle.com/cd/B10467_16/tour/websrvc_intro.htm.

[2] Protokol SOAP, 1. del. Dostopno na spletnem naslovu: http://www.w3.org/TR/2001/WD–soap12–part1–20011217/.

[3] Kratka navodila za XSD. Dostopno na spletnem naslovu: http://skaterpro.net/xsd.htm.

[4] Razumevanje UDDI–ja. Dostopno na spletnem naslovu: http://www.ibm.com/developerworks/webservices/library/ws–featuddi/.

[5] Specifikacije spletnih storitev. Dostopno na spletnem naslovu: http://msdn.microsoft.com/en–us/library/ms951274.aspx.

[6] Simple Object Access Protocol (SOAP) 1.1. Dostopno na spletnem naslovu: http://www.w3.org/TR/2007/REC–soap12–part0–20070427/#intro.

[7] Web Services Description Language (WSDL) 1.1. Dostopno na spletnem naslovu: http://www.w3.org/TR/wsdl.

[8] Tutorial za XML Scheme. Dostopno na spletnem naslovu: http://www.w3schools.com/schema/default.asp.

[9] Extensible Markup Language (XML) 1.0. Dostopno na spletnem naslovu: http://www.w3.org/TR/REC–xml/.

[10] Imenski prostori v XML. Dostopno na spletnem naslovu: http://www.w3.org/TR/REC–xml–names/.

[11] Nabor informacij v XML. Dostopno na spletnem naslovu: http://www.w3.org/TR/xml–infoset/.

[12] WSDL 1.1 Binding Extension for SOAP 1.2. Dostopno na spletnem naslovu: http://schemas.xmlsoap.org/wsdl/soap12/WSDL11SOAP12.pdf.

[13] WSDL 2.0. Dostopno na spletnem naslovu: http://www.xml.com/pub/a/ws/2004/05/19/wsdl2.html.

[14] Web Services Policy Framework (WSPolicy). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2004/09/policy/ws–policy.pdf.

[15] Web Services Policy Attachment (WSPolicyAttachment). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2004/09/policy/ws–policyattachment.pdf.

[16] Web Services Metadata Exchange (WS–MetadataExchange). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2004/09/mex/WS–MetadataExchange.pdf.

[17] Web Services Dynamic Discovery (WS–Discovery). Dostopno na spletnem naslovu: http://schemas.xmlsoap.org/ws/2004/10/discovery/ws–discovery.pdf.

[18] MTOM Serialization Policy Assertion (WS–MTOMPolicy). Dostopno na spletnem naslovu: http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization/optimizedmimeserialization–policy.pdf.

[19] Web Services Addressing (WS–Addressing). Dostopno na spletnem naslovu: http://www.w3.org/Submission/ws–addressing/.

[20] SOAP Message Transmission Optimization Mechanism. Dostopno na spletnem naslovu: http://www.w3.org/TR/soap12–mtom/.

[21] Web Services Enumeration (WS–Enumeration). Dostopno na spletnem naslovu: http://www.w3.org/Submission/WS–Enumeration/.

[22] Web Services Eventing (WS–Eventing). Dostopno na spletnem naslovu: http://www.w3.org/Submission/WS–Eventing/.

[23] Web Services Transfer (WS–Transfer). Dostopno na spletnem naslovu: http://www.w3.org/Submission/WS–Transfer/.

[24] SOAP–over–UDP. Dostopno na spletnem naslovu: http://msdn.microsoft.com/en–us/library/ms951224.aspx.

[25] SOAP 1.1 Binding for MTOM 1.0. Dostopno na spletnem naslovu: http://schemas.xmlsoap.org/soap/mtom/SOAP11MTOM10.pdf.

[26] Web Services Coordination (WS–Coordination). Dostopno na spletnem naslovu: http://msdn.microsoft.com/en–us/library/ms951231.aspx.

[27] Web Services Atomic Transaction (WS–AtomicTransaction). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2004/10/wsat/wsat.pdf.

[28] Web Services Business Activity Framework (WS–BusinessActivity). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2004/10/wsba/wsba.pdf.

[29] Web Services Reliable Messaging Protocol (WS–ReliableMessaging). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2005/02/rm/ws–reliablemessaging.pdf.

[30] Web Services Reliable Messaging Policy Assertion (WS–RM Policy). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2005/02/rm/WS–RMPolicy.pdf.

[31] Web Services Security: SOAP Message Security 1.0 (WS–Security 2004). Dostopno na spletnem naslovu: http://docs.oasis–open.org/wss/2004/01/oasis–200401–wss–soap–message–security–1.0.pdf.

[32] Web Services Security UsernameToken Profile 1.0. Dostopno na spletnem naslovu: http://docs.oasis–open.org/wss/2004/01/oasis–200401–wss–username–token–profile–1.0.pdf.

[33] Web Services Security X.509 Certificate Token Profile. Dostopno na spletnem naslovu: http://docs.oasis–open.org/wss/2004/01/oasis–200401–wss–x509–token–profile–1.0.pdf.

[34] Web Services Security Policy Language (WS–SecurityPolicy). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws–securitypolicy.pdf.

[35] Web Services Trust Language (WS–Trust). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2005/02/trust/ws–trust.pdf.

[36] Web Services Secure Conversation Language (WS–SecureConversation). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2005/02/sc/WS–SecureConversation.pdf.

[37] Web Services Federation Language (WS–Federation). Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2006/12/federation/ws–federation.pdf.

[38] WS–Federation: Active Requestor Profile. Dostopno na spletnem naslovu: http://public.dhe.ibm.com/software/dw/specs/ws–fedact/ws–fedact.pdf.

[39] WS–Federation: Passive Requestor Profile. Dostopno na spletnem naslovu: http://public.dhe.ibm.com/software/dw/specs/ws–fedpass/ws–fedpass.pdf.

[40] Web Services Security Kerberos Token Profile 1.1. Dostopno na spletnem naslovu: http://docs.oasis–open.org/wss/v1.1/wss–v1.1–spec–errata–os–KerberosTokenProfile.pdf.

[41] Web Single Sign–On Interoperability Profile. Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2005/04/ssi/WebSSOInteropProfile.pdf.

[42]Web Single Sign–On Metadata Exchange Protocol. Dostopno na spletnem naslovu: http://specs.xmlsoap.org/ws/2005/04/ssi/WebSSOMEXProtocol.pdf.

[43] WS–I Basic profile. Dostopno na spletnem naslovu: http://ws–i.org/Profiles/BasicProfile–1.2–2010–11–09.html#philosophy.

[44] Razlaga REST. Dostopno na spletnem naslovu: http://www.slideshare.net/dnene/rest–representational–state–transfer–explained.

[45] Web Application Description Language. Dostopno na spletnem naslovu http://www.w3.org/Submission/wadl/.

[46] Uvod v JSON. Dostopno na spletnem naslovu: http://json.org/json–sl.html.

[47] SOAP vs. REST. Dostopno na spletnem naslovu: http://www.youtube.com/watch?v=v3OMEAU_4HIXxx.

[48] SOAP vs. REST: Complements or Competitors? Dostopno na spletnem naslovu: http://proceedings.esri.com/library/userconf/devsummit09/papers/keynote_chappell.pdf.

[49] The Java API for XML–Based Web Services (JAX–WS) 2.2. Dostopno na spletnem naslovu: http://download.oracle.com/otndocs/jcp/jaxws–2.2–mrel3–evalu–oth–JSPec/.

[50] Java API for XML Processing (JAXP) 1.4. Dostopno na spletnem naslovu: http://jaxp.java.net/docs/spec/html/.

[51] Opis StAX in primerjava. Dostopno na spletnem naslovu: http://wso2.org/library/1844.

[52] JAXM . Dostopno na spletnem naslovu: http://docs.oracle.com/javaee/1.3/tutorial/doc/IntroWS6.html.

[53] Revizija vzdrževanja za Java API for XML Messaging, version 1.1. Dostopno na spletnem naslovu: http://www.jcp.org/aboutJava/communityprocess/maintenance/jsr067/jaxm–changelog–accepted.html.

[54] Web Services for Java EE, verzija 1.3. Dostopno na spletnem naslovu: http://download.oracle.com/otn–pub/jcp/websvcs–1.3–mrel2–evaluate–oth–JSPec/websvcs–1_3–final–spec.pdf.

[55] SOAP with Attachments API for Java (SAAJ) 1.3. Dostopno na spletnem naslovu: http://java.sun.com/xml/downloads/saaj.html.

[56] Java API for XML–based RPC JAX–RPC 1.1. Dostopno na spletnem naslovu: http://download.oracle.com/otndocs/jcp/7922–jax_rpc–1.1–fr–spec–oth–JSPec/.

[57] Nadgradnja platforme Java EE 6. Dostopno na spletnem naslovu: http://weblogs.java.net/blog/robc/archive/2007/09/java_ee_6_platf.html.

[58] Java API for XML Registries (JAXR). Dostopno na spletnem naslovu: http://java.sun.com/webservices/jaxr/overview.html

[59] JAX–RS: Java API for RESTful Web Services. Dostopno na spletnem naslovu: http://download.oracle.com/otn–pub/jcp/jaxrs–1.1–mrel–eval–oth–JSPec/jax_rs–1_1–mrel–spec.pdf.

[60] The Java Architecture for XML Binding (JAXB) 2.0. Dostopno na spletnem naslovu: http://jcp.org/aboutJava/communityprocess/final/jsr222/index.html.

[61] Kaj je WSIT. Dostopno na spletnem naslovu: http://metro.java.net/guide/What_is_WSIT_.html.

[62] Java XML Digital Signature API Specification (JSR 105). Dostopno na spletnem naslovu: http://docs.oracle.com/javase/6/docs/technotes/guides/security/xmldsig/XMLDigitalSignature.html#wp268799.

[63] Java Persistence API. Dostopno na spletnem naslovu: http://download.oracle.com/otn–pub/jcp/ejb–3_0–fr–eval–oth–JSPec/ejb–3_0–fr–spec–persistence.pdf.

[64] Projekt Metro. Dostopno na spletnem naslovu: http://metro.java.net/.

[65] Projekt Jersey. Dostopno na spletnem naslovu: http://jersey.java.net/.

[66] Pojasnilo pojma »deprecated«. Dostopno na spletnem naslovu: http://en.wikipedia.org/wiki/Deprecation.

[67] Vrste medijev za JAX–RS. Dostopno na spletnem naslovu: http://jackson.codehaus.org/javadoc/jax–rs/1.0/javax/ws/rs/core/MediaType.html.

[68] Orodje NetBeans. Dostopno na spletnem naslovu: http://netbeans.org/features/index.html.

[69] Aplikacijski strežnik GlassFish. Dostopno na spletnem naslovu: http://glassfish.java.net/public/getstarted.html.

[70] Orodje WampServer. Dostopno na spletnem naslovu: http://www.wampserver.com/en/.

[71] Razširitev za FireFox RESTClient. Dostopno na spletnem naslovu: https://addons.mozilla.org/en–US/firefox/addon/restclient/.

[72] WS–I Deliverables. Dostopno na spletnem naslovu: http://www.ws–i.org/deliverables/Default.aspx.

Slike

Tabele