Leksioni 9
REST & SOAP concept
JAX-WS-Document Style
SOAP-handler
Spring REST

9.1 Java API për Shërbimet e Bazuara në XML

(JAX-WS) është modeli i programimit të shërbimeve të brezit të ri që komplimenton themelin e ofruar nga Java API për modelin e programimit RPC (JAX-RPC) të bazuar në XML. Duke përdorur JAX-WS, zhvillimi i shërbimeve në internet dhe klientëve thjeshtësohet me më shumë pavarësi në platformë për aplikacionet Java duke përdorur proxi dinamike dhe shënime Java.

JAX-WS është një model programimi që thjeshton zhvillimin e aplikacioneve përmes mbështetjes së një modeli standard, të bazuar në shënime për të zhvilluar aplikacione të shërbimit të internetit dhe klientëve. Teknologjia JAX-WS strategjikisht përputhet me trendin aktual të industrisë drejt një modeli mesazhesh, duke patur në qendër dokumentet dhe zëvendëson modelin e programimit të thirrjeve të procedurave të gjata siç përcaktohet nga JAX-RPC. Ndërsa modeli dhe aplikacionet e programimit JAX-RPC mbështeten ende nga ky produkt, JAX-RPC ka kufizime dhe nuk mbështet shërbime të ndryshme komplekse të përqendruara në dokument. JAX-WS është modeli strategjik i programimit për zhvillimin e shërbimeve në internet dhe është një pjesë e kërkuar e Java Platform, Enterprise Edition 6 (Java EE 6). JAX-WS njihet gjithashtu si JSR 224.

Versioni 8.0 mbështet specifikimet e Versionit 1.3 të JAX-WS Version 2.2 dhe Shërbime Web për Java EE (JSR 109).

Specifikimi JAX-WS 2.2 zëvendëson dhe përfshin funksione brenda specifikimit JAX-WS 2.1. JAX-WS 2.2 shton mbështetjen nga ana e klientit për përdorimin e shënimeve të lidhura me WebServiceFeature si @MTOM, @Addressing dhe shënimet @RespectBinding. JAX-WS 2.1 më parë kishte shtuar mbështetjen për këto shënime në server. Tani ekziston gjithashtu aftësia për të mundësuar dhe konfiguruar mbështetjen e Adresimit WS në një klient ose shërbim duke shtuar pohimet e Politikës WS në dokumentin WSDL. Përveç kësaj, specifikimet e Shërbimeve të Webit për Java EE 1.3 paraqesin mbështetje për këto shënime të lidhura me WebServiceFeature, si dhe mbështetje për përdorimin e elementeve përshkruese të vendosjes për të konfiguruar këto karakteristika si në klient ashtu edhe në server. JAX-WS 2.2 kërkon Java Architecture for XML Binding (JAXB) Version 2.2 për lidhjen e të dhënave.

Zbatimi i standardit të programimit JAX-WS ofron përmirësimet e mëposhtme për zhvillimin e shërbimeve të internetit dhe klientëve:

1. Pavarësia e zgjeruar e platformës për aplikacionet Java.
Duke përdorur API-të JAX-WS, zhvillimi i shërbimeve të internetit dhe klientëve thjeshtësohet me pavarësinë e zgjeruar të platformës për aplikacionet Java. JAX-WS përfiton nga mekanizmi dinamik i përfaqësimit për të siguruar një model zyrtar të delegimit me një ofrues të pluggable. Ky është një përmirësim i JAX-RPC, i cili mbështetet në gjenerimin e kërkesave specifike të shitësve për thirrje.

2. Shënime

JAX-WS ofron mbështetje për shënimin e klasave Java me metoda të dhëna për të treguar që klasa Java është një shërbim Web. JAX-WS mbështet përdorimin e shënimeve bazuar në specifikimin e Lehtësisë së Metadatave për Gjuhën e Programimit Java (JSR 175), specifikimin e Metadatës së Shërbimeve të Uebit për Platformën Java (JSR 181) dhe shënimet e përcaktuara nga specifikimi JAX-WS 2.2. Përdorimi i shënimeve brenda burimit Java dhe brenda klasës Java thjeshton zhvillimin e shërbimeve në internet. Përdorni shënimet për të përcaktuar informacionin që specifikohet zakonisht në skedarët përshkrues të vendosjes, skedarët WSDL ose në hartëzimin e meta të dhënave nga skedarët XML dhe WSDL në artefaktet burimore. Për shembull, mund të vendosni një etiketë të thjeshtë @WebService në burimin Java për të ekspozuar kodin si një shërbim në internet.

```
@WebService

public class QuoteBean implements StockQuote {

    public float getQuote(String sym) { ... }

}
```

Shënimi @WebService i thotë mjedisit të kohës së ekzekutimit të serverit të ekspozojë të gjitha metodat publike në atë kod si një shërbim në internet. Nivelet shtesë të grimcimit mund të kontrollohen duke shtuar shënime shtesë në metodat ose parametrat individualë. Përdorimi i shënimeve e bën shumë më të lehtë ekspozimin e objekteve Java si shërbime në internet. Përveç kësaj, ndërsa artefaktet krijohen nga përdorimi i disa prej mjeteve të hartës nga lart-poshtë duke filluar nga një skedar WSDL, shënimet përfshihen brenda klasës burim dhe Java si një mënyrë për të kapur meta të dhënat së bashku me skedarët burim.

Përdorimi i shënimeve gjithashtu përmirëson zhvillimin e shërbimeve në internet brenda një strukture ekipi, sepse nuk keni nevojë të përcaktoni çdo shërbim të internetit në një përshkrues të vetëm ose të zakonshëm të vendosjes siç kërkohet me shërbimet e internetit JAX-RPC. Përfitimi i shënimeve me shërbimet e internetit JAX-WS mundëson zhvillimin paralel të shërbimit dhe të meta të dhënave të kërkuara.

Për shërbimet e internetit JAX-WS, përdorimi i përshkruesit të vendosjes webservices.xml është opsional sepse mund të përdorni shënime për të specifikuar të gjithë informacionin që përmbahet në skedarin përshkrues të vendosjes. Ju mund të përdorni skedarin përshkrues të vendosjes për të shtuar ose tejkaluar shënimet ekzistuese të JAX-WS. Çdo informacion që ju përcaktoni në përshkruesin e vendosjes webservices.xml mbizotëron çdo informacion përkatës që specifikohet nga shënimet.

Për shembull, nëse klasa e zbatimit të shërbimit tuaj për shërbimin tuaj të internetit JAX-WS përfshin sa vijon:

- @WebService(wsdlLocation=”http://myhost.com/location/of/the/wsdl/ExampleService.wsdl”)
- the webservices.xml file specifies a different file name for the WSDL document as follows:
- <webservices>
- <webservice-description>
- <webservice-description-name>ExampleService</webservice-description-name>

- <wsdl-file>META-INF/wsdl/ExampleService.wsdl</wsdl-file>
- ...
- </webservice-description>
  </webservices>

Në këtë rast, vlera që specifikohet në përshkruesin e vendosjes, META-INF / wsdl / ExampleService.wsdl tejkalon vlerën e shënimit.

3. Duke thirrur shërbime në internet në mënyrë asinkrone

Me JAX-WS, shërbimet e Webit quhen sinkrone dhe asinkrone. JAX-WS shton mbështetje si për një mekanizëm të sondazhit ashtu edhe për kthimin e telefonatës kur telefononi shërbimet në internet në mënyrë asinkrone. Duke përdorur një model të sondazhit, një klient mund të lëshojë një kërkesë që një objekt të marrë përgjigjeje, i cili anketohet për të përcaktuar nëse serveri është përgjigjur apo jo. Kur serveri përgjigjet, përgjigja aktuale merret. Duke përdorur modelin e kthimit të thirrjes, klienti siguron një mbajtës të thirrjes për të pranuar dhe përpunuar objektin e përgjigjes përbrenda. Të dy modelet e sondazhit dhe kthimit të telefonatës i mundësojnë klientit të përqendrohet në vazhdimin e përpunimit të punës pa pritur për një përgjigje për t'u kthyer, ndërsa siguron një model më dinamik dhe efikas për të thirrur shërbimet e Internetit.

Për shembull, një ndërfaqe e shërbimit të webit mund të ketë metoda për kërkesa sinkrone dhe asinkrone. Kërkesat asinkrone identifikohen me shkronja të zeza në shembullin vijues:

```
@WebService
public interface CreditRatingService {
    // sync operation
    Score      getCreditScore(Customer customer);
    // async operation with polling
    Response<Score> getCreditScoreAsync(Customer customer);
    // async operation with callback
    Future<?> getCreditScoreAsync(Customer customer,
      AsyncHandler<Score> handler);
}
```

Thirrja asinkrone që përdor mekanizmin e kthimit të thirrjes kërkon një hyrje shtesë nga programuesi i klientit. Thirrja është një objekt që përmban kodin e aplikacionit që ekzekutohet kur merret një përgjigje asinkrone. Përdorni shembullin e mëposhtëm të kodit për të thirrur një mbajtës thirrjeje asinkrone:

```
CreditRatingService svc = ...;

Future<?> invocation = svc.getCreditScoreAsync(customerFred,
            new AsyncHandler<Score>() {
              public void handleResponse (
                 Response<Score> response)
              {
                Score score = response.get();
                // do work here...
              }
  }
);
```

Përdorni shembullin e mëposhtëm të kodit për të thirrur një klient sondazhi asinkron:

```
CreditRatingService svc = ...;
Response<Score> response = svc.getCreditScoreAsync(customerFred);

while (!response.isDone()) {
                    // Complete an action while we wait.
}

// No cast needed, because of generics.
Score score = response.get();
```

9.1 Përdorimi i injektimit të burimeve

JAX-WS mbështet implementimin e burimeve për të thjeshtuar më tej zhvillimin e shërbimeve në internet. JAX-WS përdor këtë tipar kryesor të Java EE 5 për të zhvendosur barrën e krijimit dhe inicializimit të burimeve të përbashkëta në një mjedis të ekzekutimit Java nga aplikacioni juaj i shërbimit të internetit në mjedisin e aplikacionit tëvetë. JAX-WS ofron mbështetje për një nëngrup të shënimeve që përcaktohen në JSR-250 për injektimin e burimeve dhe ciklin jetësor të aplikimit në mjedisin e tij të ekzekutimit.
Serveri i aplikacionit gjithashtu mbështet përdorimin e shënimit @Resource ose @WebServiceRef për të deklaruar klientët e menaxhuar nga JAX-WS dhe për të kërkuar injektimin e shërbimeve dhe porteve JAX-WS. Kur secila prej këtyre shënimeve përdoret në një fushë ose metodë, ato rezultojnë në injektimin e një shërbimi JAX-WS . Përdorimi i këtyre shënimeve gjithashtu rezulton që lloji i specifikuar nga shënimi të lidhet në hapësirën e emrave JNDI.

Shënimi @Resource përcaktohet nga specifikimi JSR-250, Shënime të Përbashkëta që përfshihet në Java Platform, Enterprise Edition 5 (Java EE 5). Duke vendosur shënimin @Resource në një ndryshore të tipit *javax.xml.ws.WebServiceContext* brenda një klase të zbatimit të pikës fundore të shërbimit, ju mund të kërkoni një injeksion burimesh dhe të mblidhni ndërfaqen *javax.xml.ws.WebServiceContext* në lidhje me atë thirrje të veçantë të pikës fundore. Nga ndërfaqja WebServiceContext, ju mund të grumbulloni MessageContext për kërkesën që lidhet me thirrjen e metodës së veçantë duke përdorur metodën *getMessageContext ().*

Shënimi @WebServiceRef përcaktohet nga specifikimi JAX-WS.

Shembulli i mëposhtëm ilustron përdorimin e shënimeve @Resource dhe @WebServiceRef për injektimin e burimeve:

```
@WebService
public class MyService {

   @Resource
   private WebServiceContext ctx;

   @Resource
   private SampleService svc;

   @WebServiceRef
```

```
    private SamplePort port;

    public String echo (String input) {
        …
    }

}
```

Lidhja e të dhënave me JAXB 2.2 JAX-WS shfrytëzon ***Java Architecture for XML Binding*** (JAXB) 2.2 API dhe mjetet si teknologji lidhëse për hartëzimet ndërmjet objekteve Java dhe dokumenteve XML. Opsionet JAX-WS mbështeten në opsionet JAXB për lidhjen e të dhënave të parazgjedhura për hartat dykahëshe midis objekteve Java dhe dokumenteve XML. Lidhja e të dhënave JAXB zëvendëson lidhjen e të dhënave të përshkruara nga specifikimi JAX-RPC. JAX-WS 2.2 kërkon JAXB 2.2 për lidhjen e të dhënave. JAXB 2.2 ofron përmirësime të vogla në shënimet e tij për gjenerim të përmirësuar të skemës dhe integrim më të mirë me JAX-WS.

9.3 Klientë dinamikë dhe statikë

Klienti dinamik API për JAX-WS quhet klienti i dërgimit (javax.xml.ws.Dispatch). Klienti iidërgimit është një klient i orientuar drejt mesazheve XML. Të dhënat dërgohen në mënyrën PAYLOAD ose MESSAGE. Kur përdorni mënyrën PAYLOAD, klienti i dërgimit është përgjegjës vetëm për sigurimin e përmbajtjes së <soap: Bady> dhe JAX-WS shton elementet <soap:Envelope> dhe <sapun: Header>. Kur përdorni mënyrën MESSAGE, klienti i dërgimit është përgjegjës për sigurimin e të gjithë Envelope SOAP duke përfshirë elementët <soap:Envelope >, <sapun: Header> dhe <sapun: Body>. JAX-WS nuk shton asgjë shtesë në mesazh. Klienti i dërgimit mbështet thirrje asinkrone duke përdorur një mekanizëm thirrjeje ose sondazhi. Modeli i programimit statik të klientit për JAX-WS quhet klienti i përfaqësimit. Klienti i përfaqësimit thërret një shërbim në internet të bazuar në një ndërfaqe të Shërbimit Endpoint (SEI), e cila duhet të sigurohet.

Mbështetje për MTOM

Duke përdorur JAX-WS, mund të dërgoni bashkëngjitje binare të tilla si imazhe ose skedarë së bashku me kërkesat e shërbimeve në internet. JAX-WS shton mbështetje për transmetimin e optimizuar të të dhënave binare siç specifikohet nga Mekanizmi i Optimizimit të Transmetimit të Mesazhit (MTOM).

9.3.1 Teknologji të shumta të detyrueshme të të dhënave

JAX-WS ekspozon teknologjitë e mëposhtme të lidhjes për përdoruesit përfundimtar: Burimi XML, SOAP Attachments API për Java (SAAJ) 1.3 dhe Java Arkitektura për Lidhjen XML (JAXB) 2.2. Burimi XML mundëson që një përdorues të kalojë një javax.xml.transform.Source në mjedisin e kohës së ekzekutimit që përfaqëson të dhënat në një objekt Burimi që do të përpunohen. SAAJ 1.3 tani ka aftësinë të kalojë një dokument të tërë SOAP nëpër ndërfaqe sesa vetëm ngarkesën vetë. Ky veprim bëhet nga klienti duke kaluar objektin SAAJ SOAPMessage nëpër ndërfaqe. JAX-WS shfrytëzon mbështetjen JAXB 2.2 si teknologji e zgjedhur për lidhjen e të dhënave midis Java dhe XML.

Mbështetje për SOAP 1.2

Mbështetja për SOAP 1.2 është shtuar në JAX-WS 2.0. JAX-WS mbështet të dy SOAP 1.1 dhe SOAP 1.2 në mënyrë që të mund të dërgoni bashkëngjitje binare si imazhe ose skedarë së bashku me kërkesat e shërbimeve në internet. JAX-WS shton mbështetje për transmetimin e optimizuar të të dhënave binare siç specifikohet nga MTOM.

9.3.2 Mjetet e zhvillimit

JAX-WS siguron mjetet e rreshtit të komandave wsgen dhe wsimport për gjenerimin e objekteve portative për shërbimet e internetit JAX-WS. Kur krijoni shërbime të internetit JAX-WS, mund të filloni ose me një skedar WSDL ose me një klasë të implementimit. Nëse filloni me një klasë primare implementimi, përdorni mjetin e rreshtit të komandave wsgen për të gjeneruar të gjitha artefaktet e serverit të shërbimeve të internetit, përfshirë një skedar WSDL nëse kërkohet. Nëse filloni me një skedar WSDL, përdorni mjetin e rreshtit komandues wsimport për të gjeneruar të gjitha artefaktet e shërbimeve të internetit ose për serverin ose për klientin. Mjeti i rreshtit të komandave wsimport përpunon skedarin WSDL me përkufizime të skemës për të gjeneruar objekte portative, të cilat përfshijnë klasën e shërbimit, klasën e ndërfaqes së pikës fundore të shërbimit dhe klasat JAXB 2.2 për skemën përkatëse XML.

9.4 Mbështetje për Shërbime Web për Java EE, version 1.3

Specifikimet e Shërbimeve të Wpër Java EE version 1.3 shtojnë mbështetjen për konfigurimin e veçorive MTOM, Adresimit dhe RespectBinding në shërbimet dhe klientët JAX-WS përmes përdorimit të shënimeve dhe shënimeve të përshkruesve të vendosjes.

Mbështetje për hapësirën boshe të shënjestruar për stilin e parametrave WRAPPED dhe llojet e kthimit

JAX-WS 2.2 mbështet parametrat e metodës dhe llojet e kthimit. Në një operacion të shërbimeve të internetit JAX-WS, mund të përcaktoni një operacion të shërbimeve në internet me një parametër të funksionimit dhe një lloj kthimi opsional. Nëse parametri i funksionimit dhe lloji i kthimit përcaktojnë një pronë të zbrazët të targetNames duke specifikuar një vlerë për vetinë e hapësirës namesNe me shënimin @WebParam ose @WebResult, mjedisi i ekzekutimit JAX-WS sillet në mënyrën vijuese:

Nëse operacioni është stili i dokumentit, stili i parametrit është WRAPPED, dhe parametri nuk harton në një kokë, atëherë një hapësirë bosh e emrave është e shënuar me parametrat e funksionimit dhe llojet e kthimit.

Nëse stili i parametrit nuk është i paracaktuar, atëherë përdoret vlera e parametrit targetNamespace të specifikuar duke përdorur shënimin @WebParam ose @WebResult.

[AIX Solaris HP-UX Linux Windows] Modeli i programimit të klientit JAX-WS

Java e API-së për modelin e programimit të klientit të shërbimit të internetit të bazuar në XML (JAX-WS) mbështet si API të klientit Dispatch dhe API të klientit Dynamic Proxy. Dispatch API i klientit është një model dinamik i programimit të klientit, ndërsa modeli statik i programimit të klientit për JAX-WS është klienti Dynamic Proxy. Klientët Dispatch dhe Dynamic Proxy mundësojnë thirrje sinkrone dhe asinkrone të shërbimeve të internetit JAX-WS.

1. [Shënime AIX Solaris HP-UX Linux] JAX-WS

Java API për Shërbimet e Bazuara në XML (JAX-WS) mbështetet në përdorimin e shënimeve për të specifikuar meta të dhëna të lidhura me implementimet e shërbimeve të internetit dhe për të thjeshtuar zhvillimin e shërbimeve të internetit. Shënimet përshkruajnë se si aksesohet në një zbatim të shërbimit nga ana e serverit si një shërbim në internet ose sesi një klasë Java nga ana e klientit hyn në shërbimet e internetit.

2. [AIX Solaris HP-UX Linux Windows] Paketimi i aplikacionit JAX-WS

Ju mund të paketoni një ndërfaqe programimi Java Application Program (API) për aplikacionin Shërbime Web XML (JAX-WS) si një shërbim Webi. Një shërbim webi JAX-WS përmbahet brenda një skedari arkivi të aplikacioneve në internet (LUFT) ose një moduli WAR brenda skedarit të arkivit të ndërmarrjes (EAR).

9.4 Krijimi i një mesazhi SOAP

Për të krijuar një mesazh të zbrazët SOAP, objekti SOAPMessage duhet të instancohet duke përdorur objektin MessageFactory. Më poshtë është një shembull i krijimit të një objekti SOAPMessage. Një mesazh SOAP është një shembull XML që përbëhet nga elemente dhe atribute. Për lehtësi, pjesët kryesore të një mesazhi SOAP shtypen që korrespondojnë me SAAJ. Envlope, Header dhe Body korrespondojnë përkatësisht me SOAPE enveloapz, SOAPHeader dhe SOAPBody. Lloji i SOAPElement korrespondon me elementin specifik të aplikacionit që nuk përfshihet në hapësirën e emrave SOAP.

1. SOAPPjesa SOAPTipi i envelope

SOAPPart dhe SOAPEnvelope përdoren shpesh kur punoni me mesazhet SwA (SOAP me Shtojcë). SOAPPart është niveli rrënjësor i pjesës MIME në një mesazh SwA dhe mund të arrihet duke përdorur metodën SOAPMessage.getSOAPPart ().

SOAPEnvelope arrihet përmes metodës getEnvelope () të SOAPPart.Envelope SOAPE është niveli rrënjësor i një mesazhi XML SOAP dhe përfshin metoda për të hyrë në SOAPHeader dhe SOAPBody.

2. SOAPElement Lloji

Lloji SOAPElement përdoret për të përfaqësuar drejtpërdrejt një element specifik të aplikacionit që nuk shprehet në hapësirën e emrave për SOAP 1.1 ose 1.2 XML. Ky lloj mund të përfaqësojë një element XML.

SOAPElement mund të përfshijë objekte të tjerë SOAPElement pasi një element XML mund të përmbajë elementë të tjerë XML. Lloji është lloji super i llojeve të tjerë SOAP (SOAPEnvelope, SOAPBody, SOAPBodyElement, SOAPHeader, SOAPHeaderElement dhe elementet e gabimit).

3. SOAPHloji i kokës(header)

Elementi i kokës së një mesazhi SOAP mund të ketë zero ose më shumë blloqe header.

Lloji SOAPHeader përfaqëson elementin e kokës dhe lloji SOAPHeaderElement përfaqëson secilin bllok të kokës. Lloji SOAPHeader ofron metoda për shtimin, fshirjen dhe kontrollimin e një objekti SOAPHeaderElement.

4. SOAPHeaderLloji i elementit

SOAPHeaderElement është abstraktuar nga një bllok header për të modifikuar dhe kontrolluar një atribut të një blloku të veçantë të kokës dhe elementeve të tij fëmijë. Lloji SOAPHeaderElement mund të përmbajë një ose më shumë objekte SOAPElement, si dhe atribute si 'aktori' dhe 'duhet të kuptoj'.

5. SOAPBody Type dhe SOAPBodyElement Type

Lloji SOAPBody përfaqëson elementin e trupit të një mesazhi SOAP. SOAPBodyElement shtrihet SOAPElement dhe nuk ka metoda shtesë përveç atyre që janë të trashëguara.

Më poshtë tregon një shembull të përdorimit të llojeve SOAPBody dhe SOAPBodyElement.

```
Name echo_Name = soapFactory.createName("echo", "tmax",
        "http://www.tmax.com/saaj/test");
SOAPBody body = message.getSOAPBody();
SOAPBodyElement echo_Element = body.addBodyElement(echo_Name);
```

9.5 Krijimi i mbajtësve të mesazheve SOAP

Një mbajtës i mesazheve SOAP përpunon mesazhet SOAP që shkëmbehen nga një klient JAX-RPC dhe pika e shërbimit të webit dhe mund të përdoret me një proxy të krijuar statikisht, një proxy të krijuar në mënyrë dinamike, një DII (Ndërfaqja dinamike e thirrjes), një pikë fundore e klasës Java, ose një pikë përfundimtare e EJB. Roli më i rëndësishëm i një mbajtësi të mesazheve është të sigurojë një mekanizëm për shtimin, leximin dhe trajtimin e një blloku header të një mesazhi SOAP, i cili dërgohet dhe merret nga klienti JAX-RPC dhe pika e fundit e shërbimit të webit. Përgjegjësit e mesazheve SOAP kontrollohen nga Java EE. Për të transmetuar një mesazh SOAP duke përdorur API të klientit JAX-RPC, mesazhi SOAP përpunohet përmes zinxhirit të mbajtësit të mesazheve përpara se koha e ekzekutimit të JAX-RPC të dërgohet në rrjetin e lidhur me shërbimin e internetit. Kur përgjigja merret nga një klient JAX-RPC, ajo përpunohet përmes të njëjtit zinxhir të mbajtësit të mesazheve përpara se rezultati të kthehet në programin e aplikimit të klientit. Manuali përdoret për shumë qëllime të ndryshme. Për shembull, një mbajtës i mesazheve SOAP mund të përdoret për të kriptuar mesazhet SOAP për një komunikim më të sigurt. Një klient mund të përdorë një mbajtës të mesazheve SOAP që kripton një mesazh SOAP përpara se të dërgohet. Atëherë shërbimi i synuar i webit mund të përdorë një mbajtës të mesazheve SOAP për të deshifruar mesazhin e marrë dhe ta përcjellë atë në pjesën e prapme të duhur. Përgjigja përpunohet në mënyrë të kundërt. Një shembull tjetër i përdorimit të një mbajtësi është të aksesoni informacionin nga pjesa e kokës së mesazhit SOAP. Kreu SOAP mund të përdoret për të ruajtur informacione specifike të shërbimit në internet, të cilat mund të manipulohen duke përdorur një mbajtës.

9.5 Krijimi i një Kontrolluesi Mesazhesh

Një mbajtës i mesazheve SOAP siguron një mekanizëm për përgjimin e mesazhit SOAP gjatë kërkimit dhe përpunimit të përgjigjes së shërbimit të internetit. Mund të krijoni mbajtës si nga pika e fundit e shërbimit të uebit ashtu edhe nga aplikacioni i klientit që thirret në shërbimin e webit.

Tabela e mëposhtme rendit klasat dhe ndërfaqet kryesore të API-së javax.xml.rpc.handler.

Komanda	Përshkrimi
javax.xml.rpc.handler. Handler	Ndërfaqja kryesore që përmban metoda për trajtimin e një kërkese SOAP, përgjigje dhe mesazhe faji.
javax.xml.rpc.handler. HandlerChain	Ndërfaqja HandlerChain që përfaqëson një listë të Manaxherëve. Lista ka elementin e tipit java.xml.rpc.handler.Handler.
javax.xml.rpc.handler. HandlerRegistry	Ndërfaqja HandlerRegistry që mbështet aftësinë për të rregulluar konfigurimin e mbajtësit në nivelin e programimit.
javax.xml.rpc.handler. HandlerInfo	Klasa HandlerInfo që përmban informacione në lidhje me mbajtësin e përcaktuar në skedarin webservices.xml, p.sh., parametrat e inicimit të mbajtësit.
javax.xml.rpc.handler. MessageContext	MessageContext Përmbledh kontekstin e mesazhit të përpunuar nga mbajtësi. Karakteristikat MessageContext lejojnë mbajtësin në një zinxhir mbajtës të ndajë gjendjen e tij të përpunimit.
javax.xml.rpc.handler.soap. SOAPMessageContext	Nën-ndërfaqja SOAPMessageContext e ndërfaqes MessageContext që ofron një metodë për të hyrë në kërkesën SOAP dhe mesazhet e përgjigjes
javax.xml.rpc.handler. GenericHandler	Klasa Abstrakt GenericHandler që implementon ndërfaqen Handler. Një mbajtës i mesazheve SOAP mund të trashëgojë ndërfaqen për implementim.
javax.xml.soap.SOAPMessage	Klasa e mesazhit që përmban mesazhin SOAP të kërkesës ose përgjigjes aktuale.

Më poshtë janë hapat e nivelit të lartë për shtimin e mbajtësve të mesazheve SOAP dhe zinxhirëve të mbajtësve për të azhurnuar shërbimin në internet.

1. Projektoni mbajtësit dhe zinxhirët e mbajtësit.

2. Krijoni një klasë Java për të zbatuar ndërfaqen javax.xml.rpc.handler.Handler dhe zbatoni zinxhirin e mbajtësit.

3. Përpiloni kodin Java.

4. Konfiguroni skedarin përshkrues të vendosjes së shërbimeve në internet (webservices.xml).

5. Paketoni dhe vendosni shërbimin në internet.

Një klient i shërbimit të webit gjithashtu mund të përdorë mbajtës të mesazheve SOAP.

9.6 Projektimi i një Përgjegjësi Mesazhesh dhe Një Zinxhiri Përgjegjësish

Informacioni i mëposhtëm kërkohet kur dizajnoni mbajtësit e mesazheve SOAP dhe zinxhirët e mbajtësit.

Numri i mbajtësve të nevojshëm

Sekuenca e ekzekutimeve

Nëse shërbimi në internet përfshin mbajtës të mesazheve ose komponentë të prapavijës.

Trajtuesit duhet të vendosen në skedarin webservices.xml, dhe një zinxhir mbajtës është një grup administrues, urdhri i ekzekutimit të të cilëve është fiksuar. Çdo mbajtës në një zinxhir mbajtës ka një metodë për trajtimin e kërkesës SOAP dhe një tjetër për trajtimin e përgjigjes SOAP. Kur përdorni një shërbim në internet, një shërbim i internetit JEUS ekzekuton mbajtësit duke përdorur mekanizmin e mëposhtëm.

1. Metoda handleRequest () e mbajtësve në zinxhirin e mbajtësit ekzekutohet e gjitha në rendin e specifikuar në skedarin webservices.xml.

2. Kur ekzekutohet metoda handleRequest () e mbajtësit të fundit në zinxhirin e mbajtësit, shërbimi i internetit JEUS thërret komponentin back-end që zbaton shërbimin e webit (nëse ekziston backend).

3. Kur komponenti i pjesës së prapme ka përfunduar ekzekutimin, mbajtësit në zinxhirin e mbajtësit thirren në rendin e kundërt, dhe thirret metoda handleResponse () e secilit mbajtës.

4. Kur ekzekutohet metoda handleResponse () e mbajtësit të parë të specifikuar në skedarin webservices.xml, serveri JEUS kthen përgjigjen e mesazhit SOAP tek aplikacioni i klientit që thirri shërbimin në internet.

9.5 Krijimi i një ndërfaqe të mbajtësit

Një klasë e mbajtësit të mesazheve SOAP mund të zbatohet duke zbatuar drejtpërdrejt ndërfaqen javax.xml.rpc.handler.Handler, ose trashëguar javax.xml.rpc.handler.GenericHandler që zbaton ndërfaqen javax.rpc.xml.handler.Handler.
Klasa e mbajtësit të mesazheve duhet të zbatojë metodat e mëposhtme të ndërfaqes Handler.

Komandat	Përshkrimi
init ()	Objekti HandlerInfo përmban informacion në lidhje me mbajtësin e mesazheve SOAP, si parametrat e inicializimit, të përcaktuar në webservices.xml. Përdorni metodën HandlerInfo.getHandlerConfig () për të marrë parametrat. Metoda kthen një objekt java.util.Map që përmban çifte emri-vlere. Zbatoni metodën init () për të përpunuar parametrat e inicimit të mbajtësit.
destroy()	Metoda Handler.destroy () thirret për të shkatërruar një shembull të objektit Handler. Zbatoni këtë metodë për të liruar çdo burim të fituar gjatë ciklit të jetës së mbajtësit.

getHeaders()	Metoda Handler.getHeaders () përdoret për të marrë blloqet e kokave që përpunohen nga instanca Handler.
handleRequest()	Metoda Handler.handleRequest () thirret për të përgjuar një kërkesë të mesazhit SOAP përpara se të përpunohet nga komponenti i prapavijës. Objekti MessageContext përmban përmbajtje mesazhesh që përpunohet nga mbajtësi i mesazheve SOAP. SOAPMessageContext, nën-ndërfaqja e MessageContext, mund të përdoret për të marrë ose modifikuar përmbajtjen e mesazhit të kërkesës SOAP. Përdorni SAAJ të lartpërmendur për të përpunuar mesazhin. Duke ekzekutuar SOAPMessageContext.get Message (), mund të merret një SOAPMessage që është pjesë e SAAJ API, dhe SOAPMessage përbëhet nga SOAPPart dhe bashkëngjitje. Detyra e trajtimit të mesazheve SOAP është e njëjtë me atë të programimit SAAJ.
handleResponse()	Metoda Handler.handleResponse () thirret për të përgjuar një përgjigje të mesazhit SOAP pasi të jetë përpunuar nga komponenti i prapavijës dhe para se të dërgohet përsëri tek aplikacioni i klientit që thirri shërbimin në internet. Objekti MessageContext abstragon kontekstin e mesazhit të përpunuar nga mbajtësi i mesazheve SOAP. Përdorni SOAPMessageContext, nën-ndërfaqen e MessageContext, për të marrë ose azhurnuar përmbajtjen e përgjigjes së mesazhit SOAP. Përdorni SAAJ të lartpërmendur për të përpunuar mesazhin. Një SOAPMessage, që përfshihet në SAAJ API, mund të merret duke thirrur SOAPMessageContext.get Message (), dhe SOAPMessage përbëhet nga SOAPPart dhe bashkëngjitjet. Detyra e trajtimit të mesazheve SOAP është e njëjtë me atë të programimit SAAJ.
handleFault()	Metoda Handler.handleFault () përpunon SOAP Fault bazuar në modelin e përpunimit të mesazhit SOAP. Metoda zbatohet për të trajtuar SOAP Fault të shkaktuar nga metodat handleRequest () dhe handleResponse () dhe gabimet e prapambetjes. Objekti MessageContext përmban përmbajtjen e mesazheve të përpunuara nga mbajtësi i mesazheve SOAP dhe një mesazh i përgjigjes SOAP mund të merret ose modifikohet duke përdorur SOAPMessageContext, e cila është një nën-ndërfaqe e ndërfaqes MessageContext. Përdorni SAAJ të lartpërmendur për të përpunuar mesazhin.

9.6 Konfigurimi i skedarit DD të shërbimeve në internet Java EE

Informacion i ndryshëm për mbajtësin e mesazheve SOAP dhe rendi i përpunimit të mbajtësit përcaktohet në webservices.xml. Më poshtë është një shembull i konfigurimit të informacionit të mbajtësit të mesazhit.

[Shembull] Cilësimet e Trajtuesit të Mesazheve në <port-component>

```
<port-component>
   <port-component-name>FileAttPort</port-component-name>
```

```
  . . .
  <handler>
    <handler-name>ServerAttachmentHandler</handler-name>
    <handler-class>
      filetransfer.ServerAttachmentHandler
    </handler-class>
    <init-param>
      <param-name>directory</param-name>
      <param-value>/temp</param-value>
    </init-param>
  </handler>
</port-component>
```

Përdorimi i një SOAP Message Handler nga Klienti është gjithashtu e mundur të përdorni një mbajtës të mesazheve SOAP nga klienti i shërbimeve të internetit. Përdorimi i mbajtësit të mesazheve SOAP nga klienti është identik me një shërbim në internet duke përdorur një mbajtës të mesazheve SOAP. Së pari, krijoni një klasë Java që zbaton ndërfaqen javax.xml.rpc.handler.Handler. Detyrat pas implementimit të një administruesi të klientit është e ndryshme nga ajo e implementimit të një administruesi të serverit. Dallimi është se administruesi i klientit regjistrohet drejtpërdrejt brenda programit duke përdorur klasat javax.xml.rpc.handler.HandlerInfo dhe javax.xml.rpc.handler.HandlerRegistry, në vend që të përcaktohet në DD. Referojuni shembullit në seksionin lart. ***Shembull i shkëmbimit të skedarëve midis shërbimeve në internet dhe klientëve*** Në këtë shembull, një klient i shërbimit të internetit dërgon një skedar të quajtur "File_Send.txt" te serveri i shërbimit të internetit. Shërbimi në internet e ruan skedarin në një dosje të caktuar dhe i dërgon një mesazh konfirmimi klientit që skedari është marrë me sukses. Një klient është krijuar në hapat e mëposhtëm. Zbatoni një mbajtës mesazhesh që merr një skedar. Zbatoni një prapavijë të shërbimit të internetit që merr një skedar dhe dërgon një mesazh konfirmimi. Konfiguroni përshkruesin e vendosjes së shërbimit të uebit dhe vendosni menjëherë dhe vendosni shërbimin në internet. Zbatoni një mbajtës nga ana e klientit. Zbatoni një klient të shërbimit të internetit. Ekzekutoni klientin. 23.2.6.1. Zbatimi i një Kontrolluesi të Mesazheve për Marrjen e Skedarëve Më poshtë është implementimi i një mbajtësi të mesazheve të shërbimit në internet që ekzekutohet para se metoda handleRequest () të dërgojë mesazhin SOAP në pjesën e prapme. Metoda handleRequest () përdor SAAJ API për të trajtuar skedarin e bashkangjitur në mesazhin SOAP. [Shembull 23.2] << ServerAttachmentHandler.java >>

```
package filetransfer;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.OutputStream;
import java.util.Iterator;
import java.util.Map;

import javax.xml.namespace.QName;
import javax.xml.rpc.JAXRPCException;
import javax.xml.rpc.handler.HandlerInfo;
import javax.xml.rpc.handler.MessageContext;
import javax.xml.rpc.handler.soap.SOAPMessageContext;
import javax.xml.soap.AttachmentPart;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPException;
import javax.xml.soap.SOAPMessage;

import javax.xml.rpc.handler.GenericHandler;
public final class ServerAttachmentHandler
    extends GenericHandler
{
    private File dir;

    private final String DIR_PROP="directory";

    public void init(HandlerInfo info) {
        super.init(info);
        Map m = info.getHandlerConfig();

        String dirName = (String) m.get(DIR_PROP);

        if (dirName == null) {
            throw new JAXRPCException("Property named: "
                + DIR_PROP + " was not found");
        }
        dir = new File(dirName);


        if (! dir.exists()) {
            if (! dir.mkdirs()) {
```

```
                throw new JAXRPCException("Unable to create directory: " + dirName);
            }}
        if (! dir.canWrite()) {
            throw new JAXRPCException(
                "Don't have write permission for " + dirName);
        }
    }
    private String getFileName(SOAPMessage request)
        throws SOAPException
    {
        SOAPBody body =
            request.getSOAPPart().getEnvelope().getBody();
        Object obj = body.getChildElements().next();
        SOAPElement opElem = (SOAPElement) obj;
        SOAPElement paramElem = (SOAPElement)opElem.getChildElements().next();
        return paramElem.getValue();
    }
    private void copyFile(InputStream is, OutputStream os)
        throws IOException
    {
        byte [] b = new byte[8192];
        int nr;
        while ((nr = is.read(b)) != -1) {
            os.write(b, 0, nr);
        }
    }
    public boolean handleRequest(MessageContext mc) {
        SOAPMessageContext ctx = (SOAPMessageContext) mc;
        SOAPMessage request = ctx.getMessage();
        if (request.countAttachments() == 0) {

throw new JAXRPCException("** Expected attachments");
        }
        try {
            Iterator it = request.getAttachments();

            while(it.hasNext()) {
                AttachmentPart part = (AttachmentPart) it.next();
                String fileName = getFileName(request);
```

```
            System.out.println("Received file named: " + fileName);

            File outFile = new File(dir, fileName);
            OutputStream os = null;
            InputStream  is = null;

            try {
                os = new FileOutputStream(outFile);
                is = part.getDataHandler().getInputStream();

                copyFile(is, os);

            } catch (IOException ioe) {

  ioe.printStackTrace();
                throw new JAXRPCException("Exception writing file " + fileName,
                ioe);
            } finally {
                try {
                    if (is != null) is.close();
                } catch (IOException ignore) {}

                try {
                    if (os != null) os.close();
                } catch (IOException ignore) {}
   }
        }
    } catch (SOAPException e) {
        e.printStackTrace();
        throw new JAXRPCException(e);
    }
    return true;
}

public QName[] getHeaders() {
    // TODO Auto-generated method stub

   return null;
}
```

```
}
```

Prezantimi

Shërbimet REST të Webit janë bërë mjeti numër një për integrimin e aplikacioneve në internet. Në thelbin e tij, REST përcakton se një sistem që përbëhet nga burime me të cilat ndërveprojnë klientët. Këto burime zbatohen në një mënyrë të drejtuar nga hipermedia. Spring MVC dhe Spring WebFlux secili ofrojnë një themel të fortë për të ndërtuar këto lloje të shërbimeve. Sidoqoftë, zbatimi edhe i parimit më të thjeshtë të shërbimeve të REST në internet për një sistem objektesh me shumë domen mund të jetë mjaft i lodhshëm dhe të rezultojë në një shumë të kodit.

Spring Data REST ndërtohet në magazinat e të dhënave Spring dhe automatikisht eksporton ato si burime REST. Ai përdor hipermedian për të lejuar klientët të gjejnë automatikisht funksionalitetin e ekspozuar nga depot dhe t'i integrojë këto burime në funksionalitetin e lidhur me bazën e hipermedias.

Fillimi

Spring Data REST është në vetvete një aplikacion Spring MVC dhe është krijuar në mënyrë të tillë që duhet të integrohet me aplikimet tuaja ekzistuese Spring MVC me pak përpjekje. Një shtresë ekzistuese (ose e ardhshme) e shërbimeve mund të funksionojë së bashku me Spring Data REST me vetëm punë të vogla shtesë.

9.7 Shtimi i të Dhënave të Spring REST në një Projekt Spring Boot

Mënyra më e thjeshtë për të filluar është të ndërtoni një aplikacion Spring Boot sepse Spring Boot ka një starter për Spring Data REST dhe përdor konfigurimin automatik. Shembulli i mëposhtëm tregon mënyrën e përdorimit të Gradle për të përfshirë Spring Data Rest në një projekt Spring Boot:

Shembull 3. Konfigurimi Spring Boot me Gradle

```
dependencies {
  ...
  compile("org.springframework.boot:spring-boot-starter-data-rest")
  ...
}
```