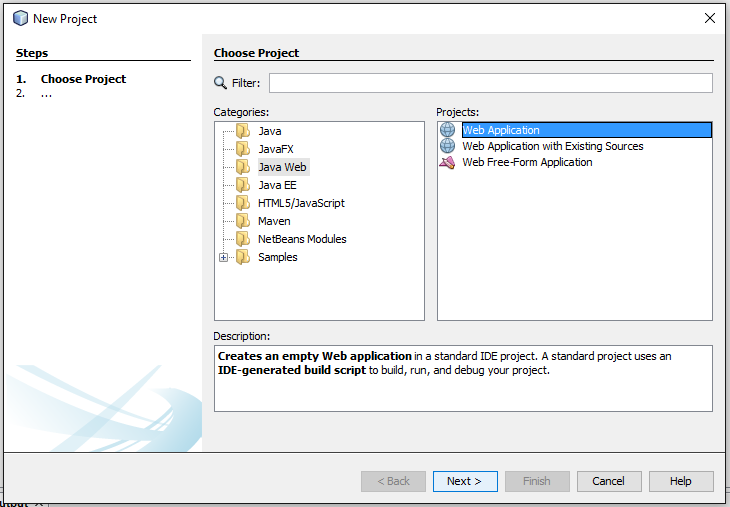
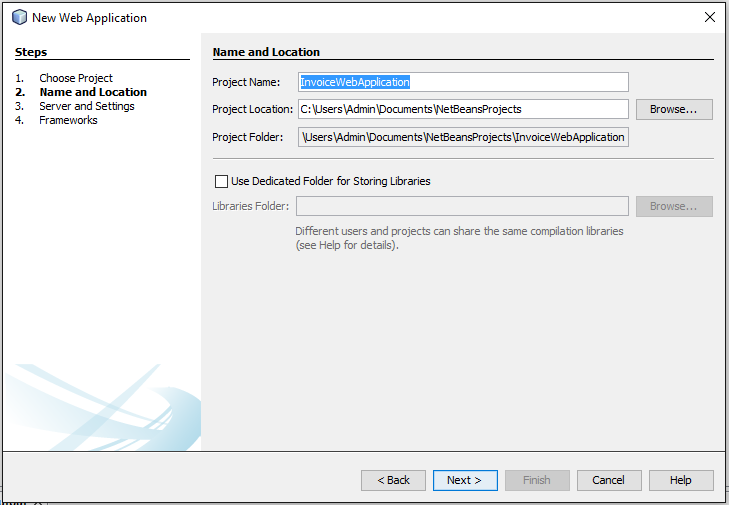
# IDU0075 Projekti loomise juhend (SOAP ja REST)

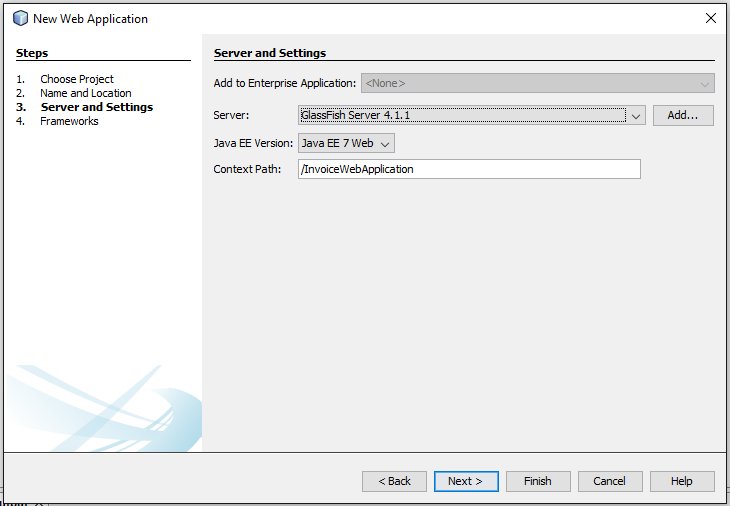
1. Loo Netbeansis Java Web Application



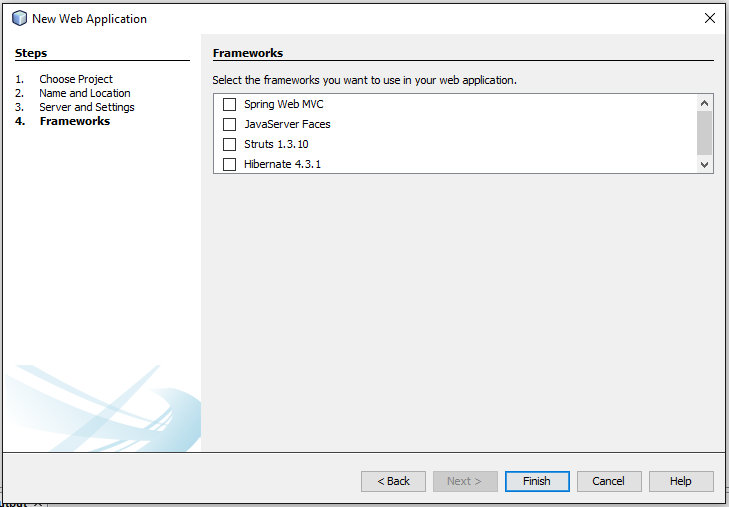
1. Määra projektile nimi



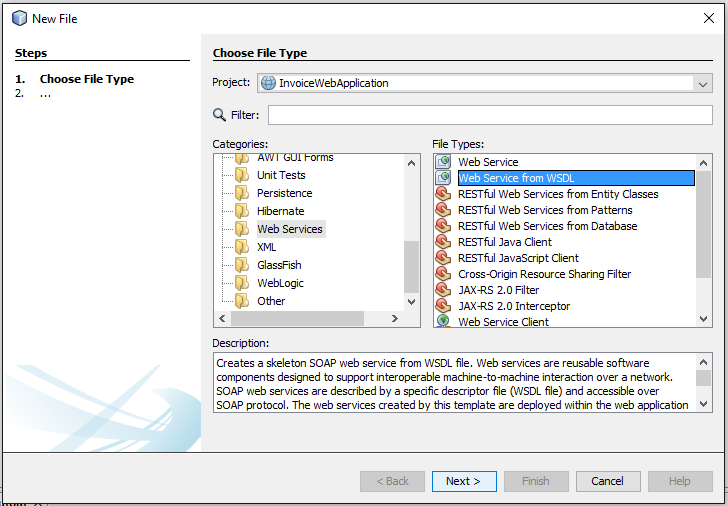
1. Jäta vaikimisi väärtused



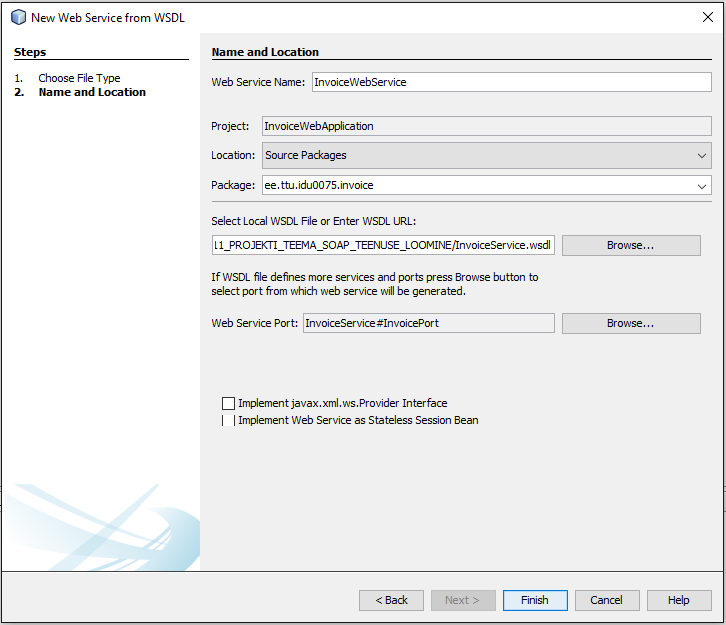
1. Raamistikke pole vaja lisada



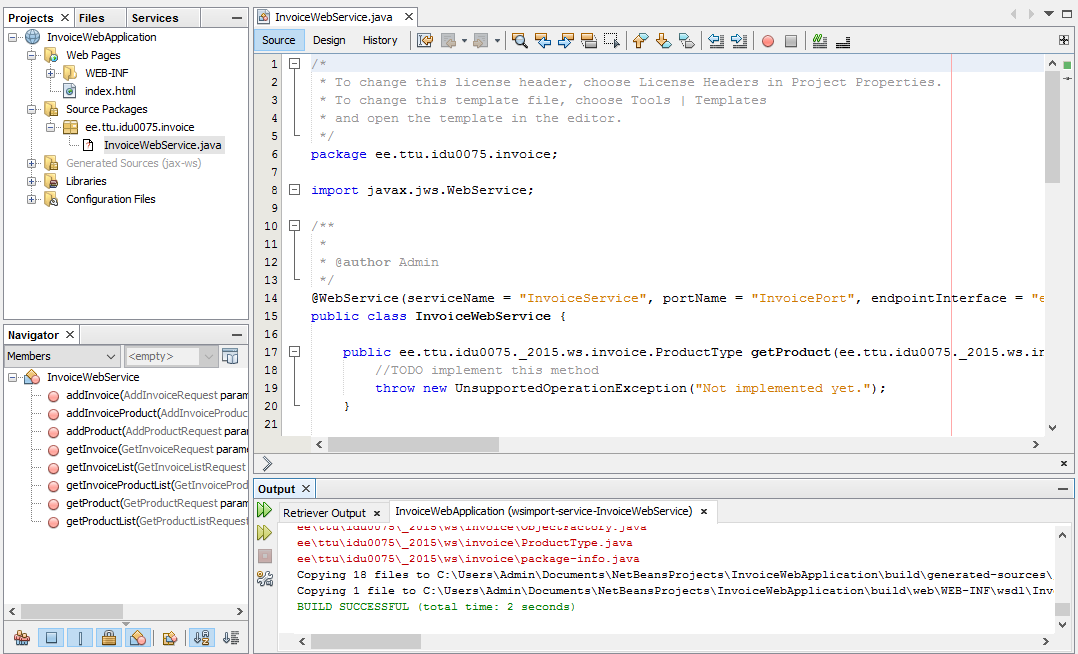
1. Kui projekt loodud, siis loo uus veebiteenus WSDL-i põhjal



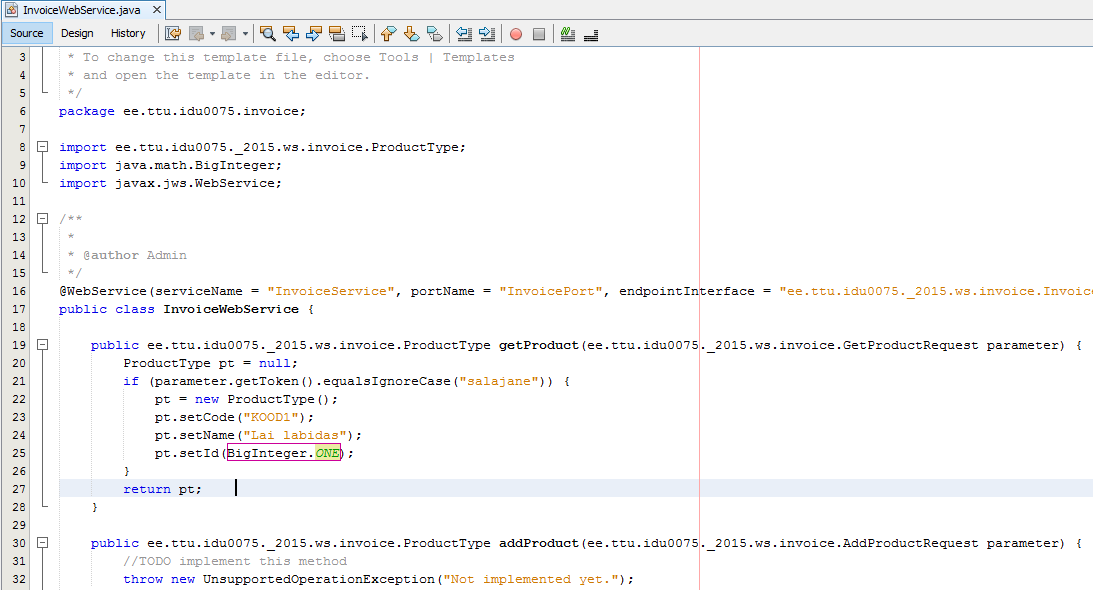
1. Määra teenusele nimi, pakett ja anna ette WSDL-i asukoht



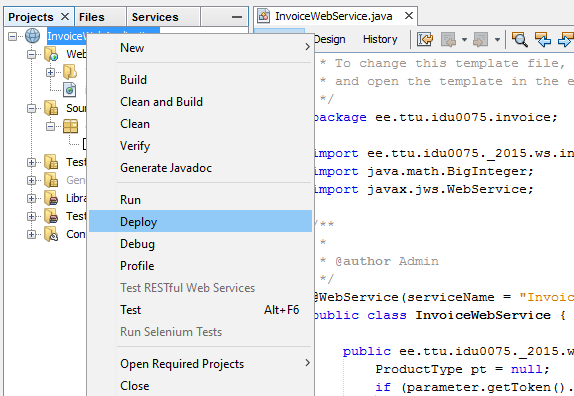
1. Kui teenuse genereerimine oli edukas, siis peaks avanema järgnevaga sarnane vaade



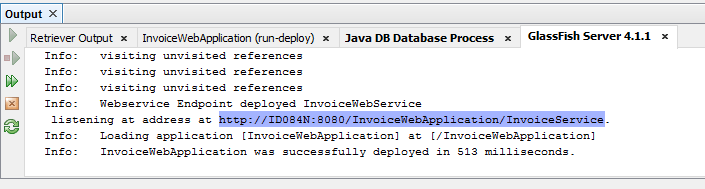
1. Alustuseks realiseerime getProduct operatsiooni tagastades sissekirjutatud väärtused, kui päringus olnud token langeb kokku sõnaga „salajane“



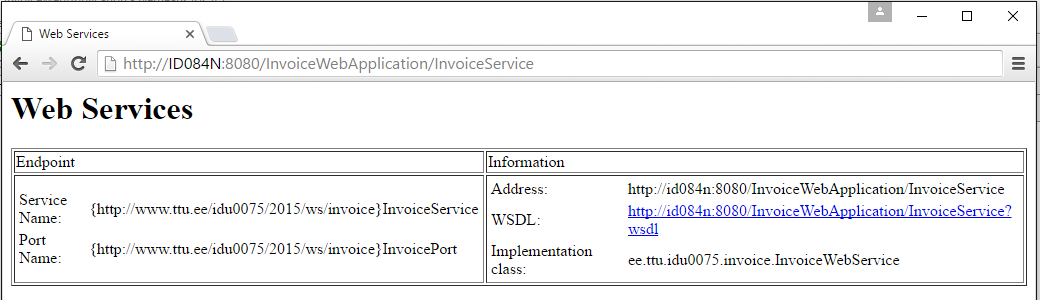
1. Esimese operatsiooni katsetamiseks laeme teenuse serverisse „deploy“ käsuga



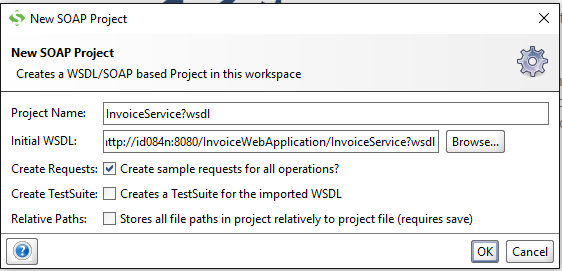
1. Kui teenuse ülesselaadimine oli edukas, siis peaks Glassfishi logisse tulema rakenduse aadress



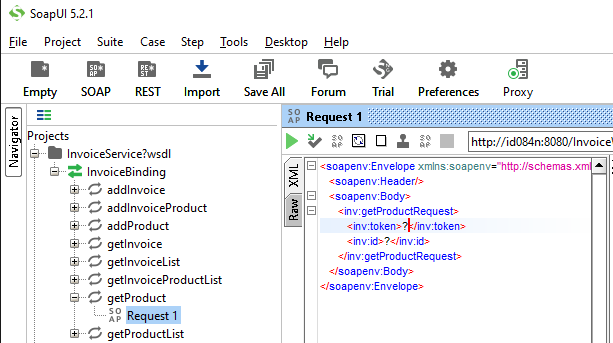
1. Rakenduse aadressile minnes saab sealt WSDL-i asukoha



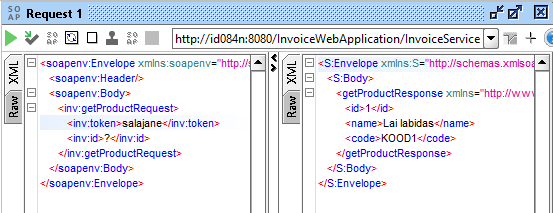
1. WSDL-i aadressi põhjal saame SoapUI-ga teha uue SOAP projekti



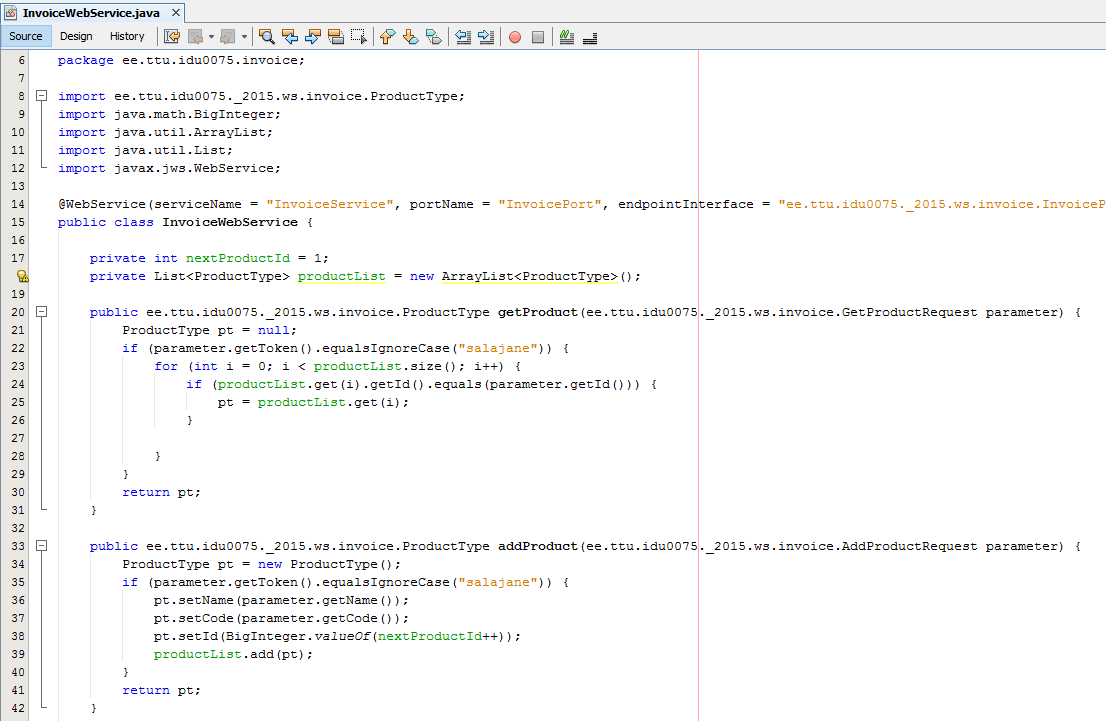
1. SOAP projekt vahetult pärast loomist



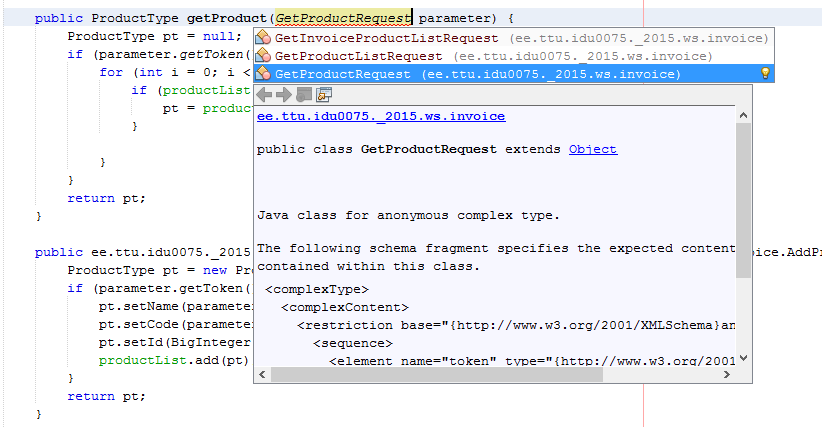
1. Saadame getProduct request-i teenusele ja same vastuseks sissekirjutatud toote



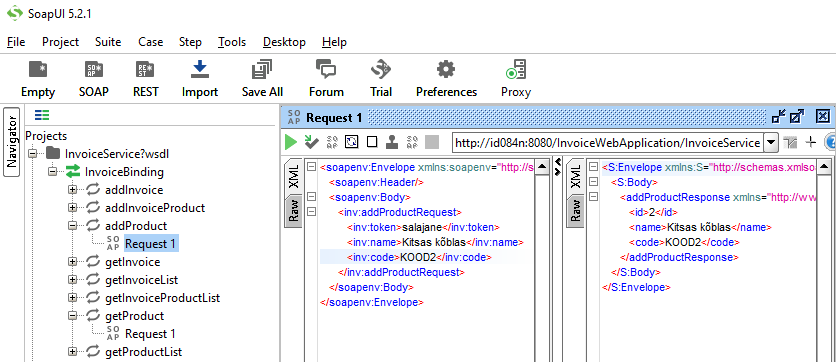
1. Muudame teenuse koodi nii, et lisatud tooted läheksid lokaalsesse listi ja toodete küsimisel otsitakse id järgi toodet listist



1. Paketti nimed võib klassi nime eest ära võtta. Sel juhul peab klassid importima



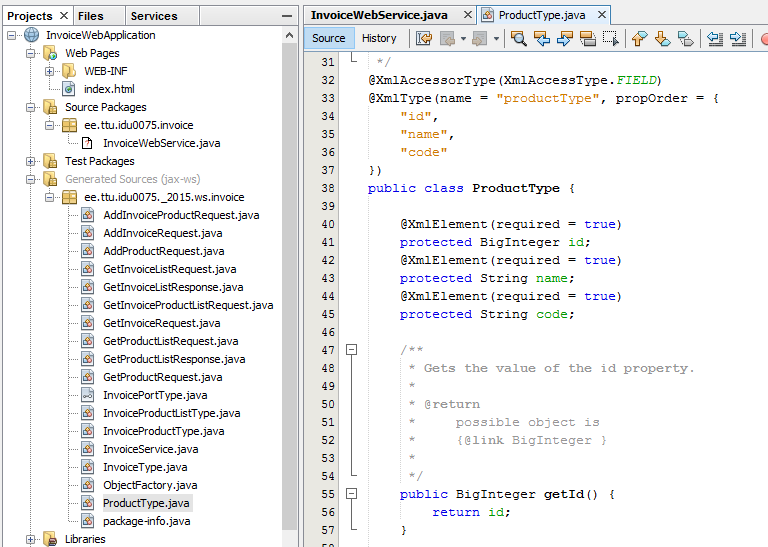
1. Proovime toote lisamist SoapUI-ga



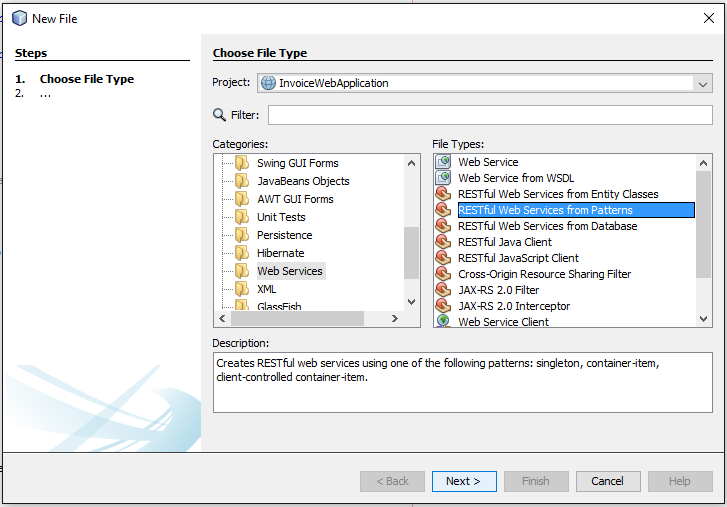
1. Realiseerime lisaks getProductList operatsiooni



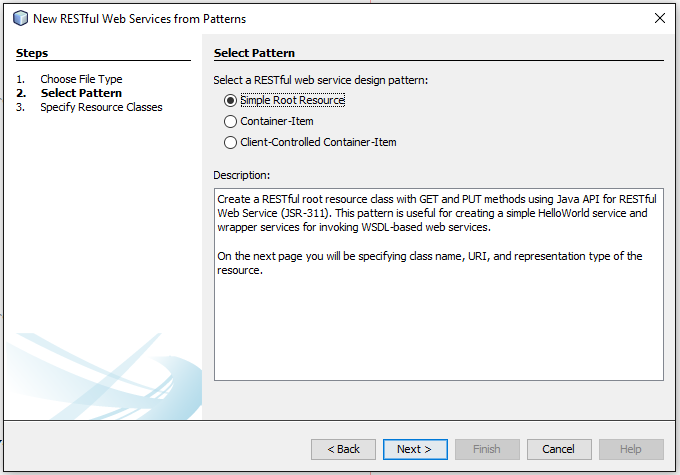
1. WSDL-i põhjal teenust tehes genereeriti jax-ws-i poolt klasse. Järgnevalt hakkame looma REST teenust, kus toote ressursi jaoks saame ära kasutada ProductType klassi



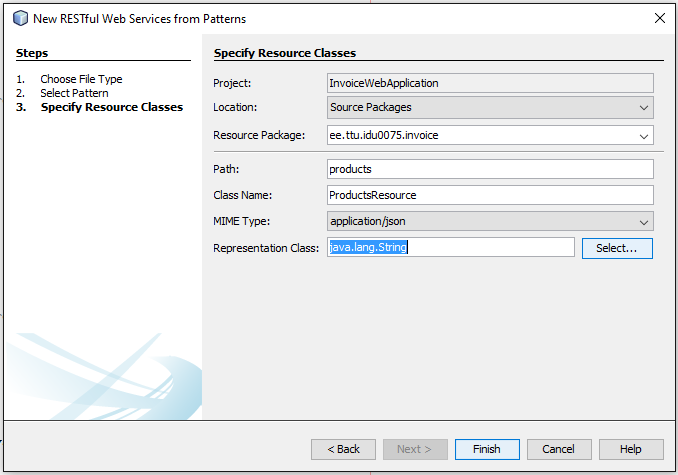
1. Loome REST teenuse mustri põhjal



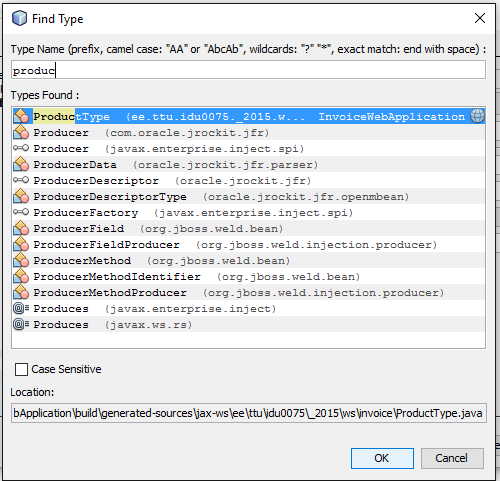
1. Mustriks valime Simple Root Resource



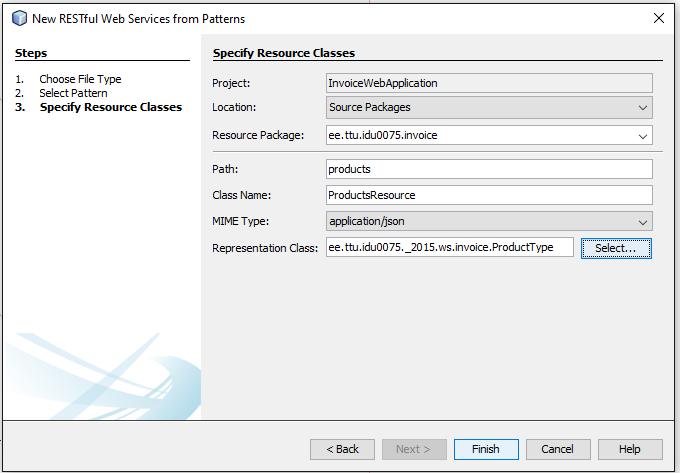
1. Määrame ressursi pathi, klassinime, MIME tüübi ja esindusklassi



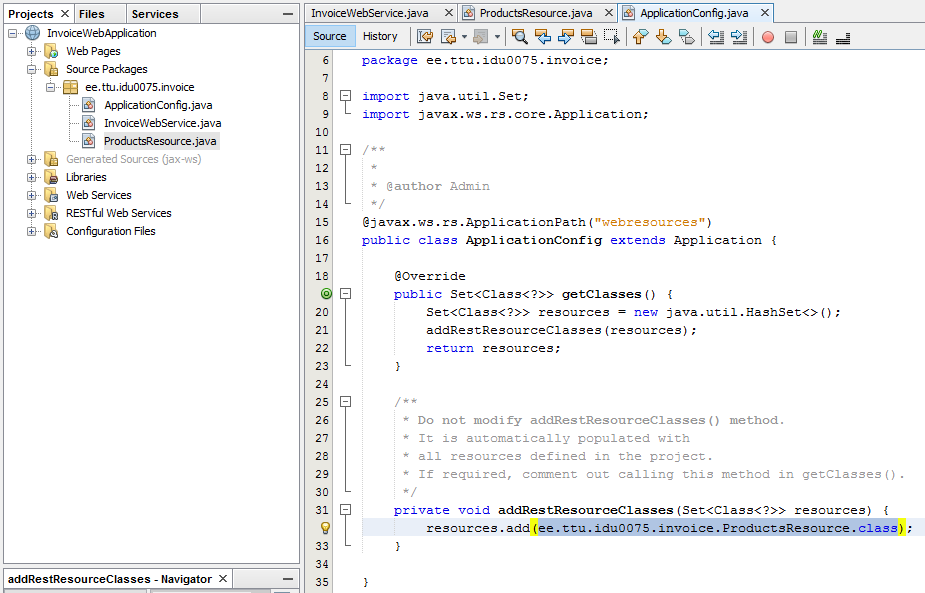
1. Esindusklassiks valime olemasoleva klassi ProductType



1. Esindusklass on valitud



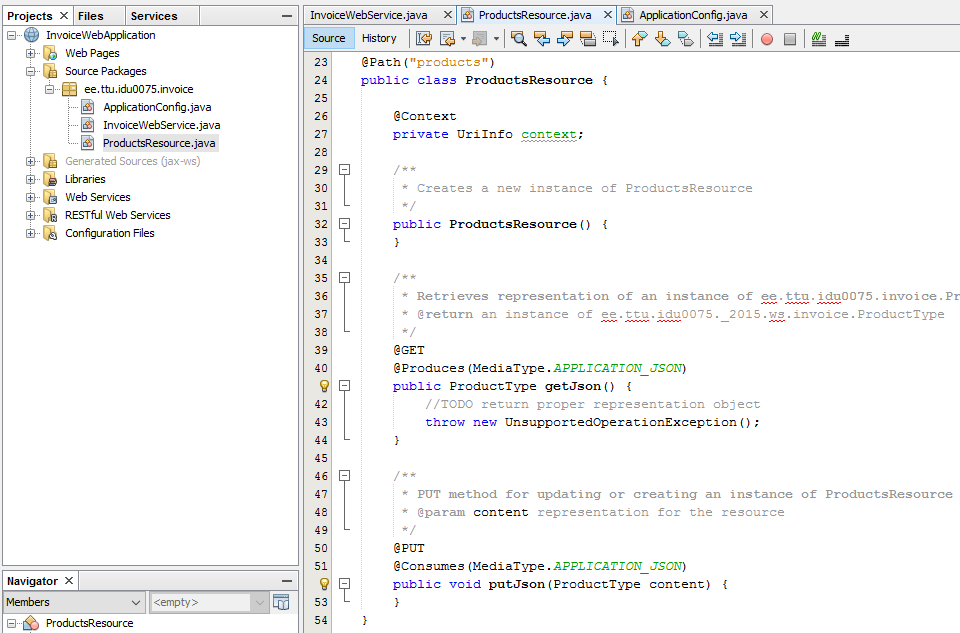
1. Genereeritakse kaks klassi ApplicationConfig ja ProductsResource. Algselt on ApplicationConfig järgmine



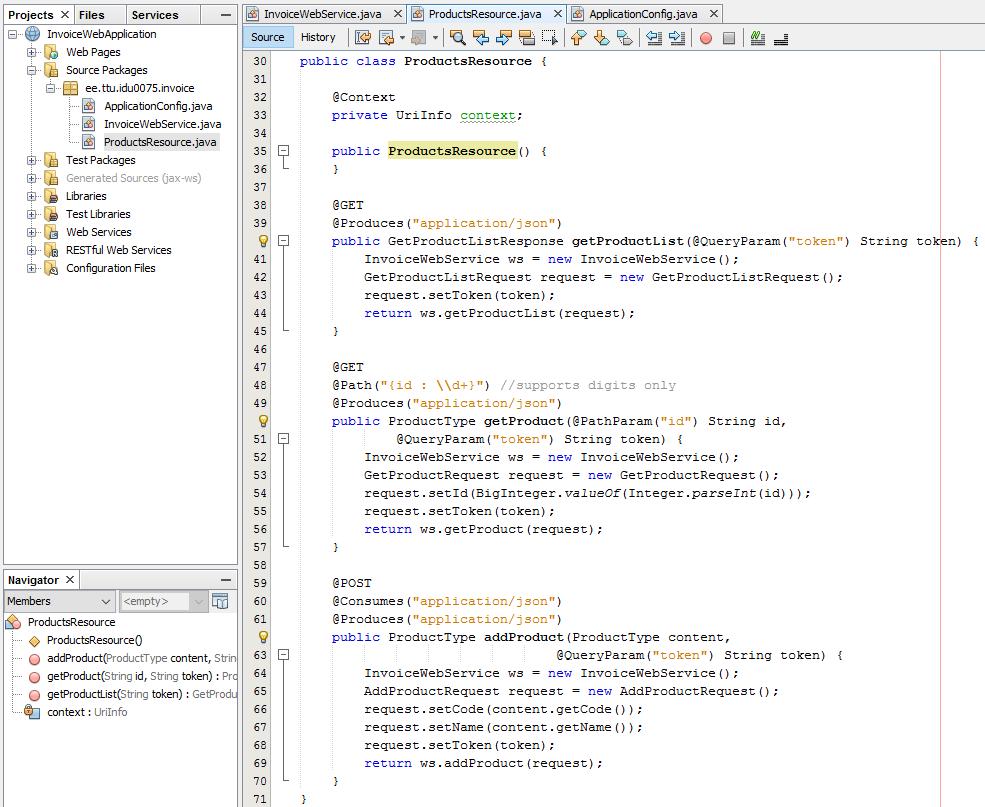
1. Kui me soovime @XmlRootElement või @XmlType abil JSON formaati konverteeritavat klassi kasutada nagu ProductType, siis tuleb lisada ApplicationConfig faili märgistatud lõik



1. Genereeritud ProductsResource näeb algselt välja järgnevalt



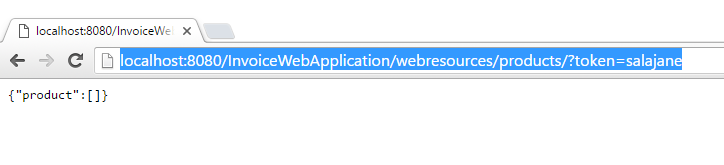
1. Näide, kuidas REST teenus on sisemiselt lahendatud SOAP teenuse jaoks loodud klassi abil



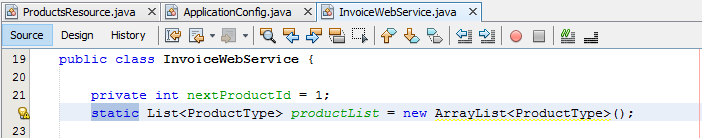
1. REST teenuse väljakutseks tuletatud URL

<http://localhost:8080/InvoiceWebApplication/webresources/products/?token=salajane>

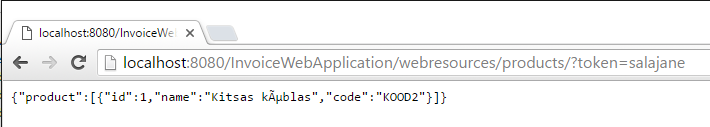
1. Get päringut saame veebilehitsejaga katsetada



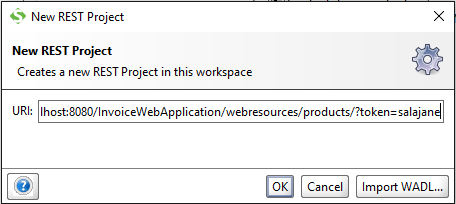
1. Kui soovime SOAP teenusega ühist toodete listi, siis tuleb teha list staatiliseks, sest SOAP ja REST jaoks tehakse InvoiceService klassist erinev objekt



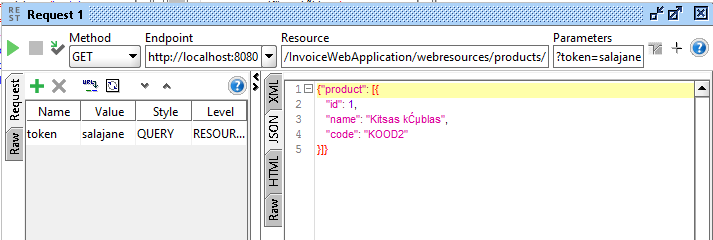
1. Kui käivitada uuesti SOAP teenuse toote lisamise näidet, siis REST-ga peaks tulemus olema järgnev



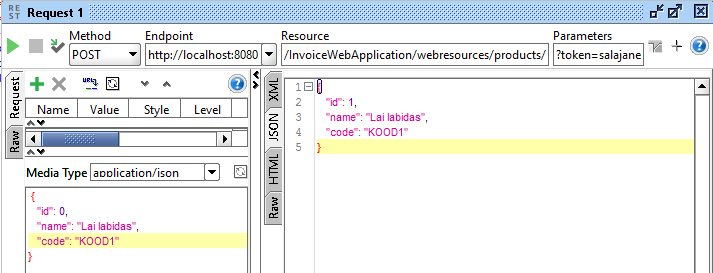
1. REST-i päringud saab saata lisaks veebilehitsejale ka SoapUI-ga. Selleks tuleb URL-i põhjal teha uus projekt



1. Toodete nimekirja REST päring SoapUI-ga



1. RESR toote lisamise katsetamine SoapUI-ga. Päringusse antakse toote id väli kaasa, kuna id oli ProductType klassis märgitud kohustuslikuks. Reaalselt genereeritakse tootele id teenuse poolel.



1. SOAP toodete nimekirja küsimise operatsiooni näide. Vastuses on nii SOAP kui ka REST abil lisatud tooted samaaegselt. Mõlema toote id-d on 1, kuna lisaks productList-le tuleb ka nextProductId muuta InvoiceWebService klassis staatiliseks

