BAB 4
Membangun Web Services

Pada proyek ini, akan dibuat modul EJB yang berisi lojik untuk menetapkan tingkat bunga untuk dua jenis pinjaman (Mobil dan Rumah). Seperti yang telah dijelaskan pada Skenario, lojik implementasi dibuat secara sederhana saja, yang dapat digunakan untuk pembuktian saja. Selanjutnya, organisasi WHUBANK merencanakan melakukan plug-in sebuah mesin aturan yang canggih pada implementasi nanti, sehingga implementasi dapat diubah, tetapi interfacenya tidak berubah. Modul EJB akan bertindak sebagai endpoint web services yang menerima input SOAP dan menghantar output SOAP.

Selanjutnya akan dibangun fungsionalitas untuk menghitung pembayaran pinjaman dengan menggunakan endpoint web service berbasis servlet. Langkah ini sangat mirip dengan pendefinisian implementasi java untuk web services, dengan menyerahkan pada IDE Netbeans untuk menciptakan XSD, WSDL dan dokumen XML.

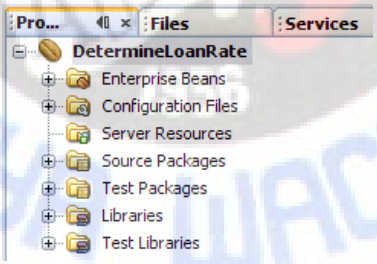
4.1 Web Service untuk Menentukan Bunga Pinjaman

Pertama kali lakukan start GlassFish ESB. Setiap proyek akan dibuat secara terpisah, sehingga akan ada lima proyek. *Start / All Programs /* GlassFish ESB pada opsi menu. Kurang dari satu menit IDE siap digunakan. Sekarang akan dimulai proyek pertama:

- Pilih *File / New Project from the menu bar*, maka wizard windows *New Project* akan tampak.
- Pilih *Java EE* dari category dan *EJB Module*
- Untuk jenis proyek. Kemudian klik *Next*.

{Catatan : Server aplikasi GlassFish Java EE ada pada bagian atas. EJB merupakan pilihan baik untuk melakukan deploy business logic seperti menentukan bunga pinjaman.}

- Untuk *Project Name*, masukkan *DetermineLoanRate*. Untuk
- *Project Location*, masukkan: *C:\LatihanSOA*. Jika direktori belum ada, maka akan secara otomatis dibuatkan. Klik *Next*.
- Terima sisanya sebagai defaults dengan melakukan klik *Finish*. Proyek *DetermineLoanRate* akan tampak.



Praktek terbaik dalam membuat web services adalah dengan mendefinisikan interfacenya sebelum membuat implementasinya. Sehingga, pertama akan dibuat dokumen XML Schema Definition (XSD) yang

mendefinisikan jenis data (*data types*) yang digunakan dalam pesan (*messages*). Kemudian membangun dokumen Web Service Description Language (WSDL) yang diimport dari XSD dan mendefinisikan penuh Web service nya.

> *{Ingat bahwa ini merupakan penyimpangan dari "best practices". Disini langsung dibuat implementasi Web service tanpa membuat interfacenya dulu.}*

Selanjutnya akan diikuti jalur dalam implementasi yang sudah disediakan, dan membiarkan Netbeans menciptakan interface (XSD dan WSDL). Pembenaran untuk melakukan shortcut ini adalah bahwa web services akan diperoleh secara cepat dan dapat dilakukan build dan testing. Kemudian dalam proyek3, ada kesempatan untuk membangun definisi interface secara benar secara bertahap dari XSD, WSDL kemudian baru Web services.

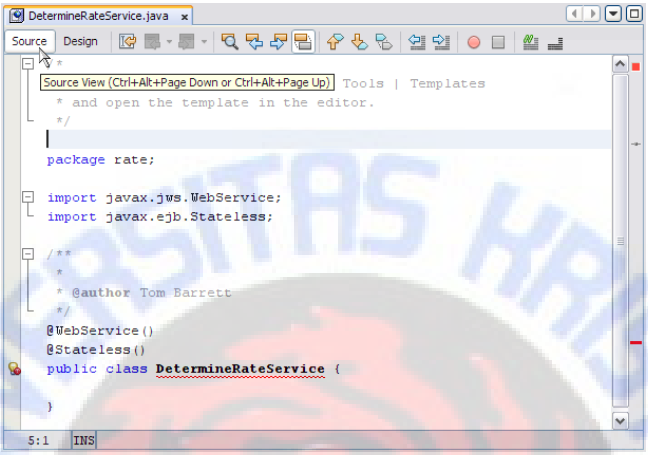
Pada node *DetermineLoanRate* pada panel *Projects* lakukan klik kanan, dan pilih *New / Web Service*. Windows wizard *New Web Service* akan tampak.

> *{Jika ini merupakan pertama kali dalam menggunakan fitur New / Web Service, mungkin perlu pergi ke File / New / Other untuk menemukan opsi Web Service. Agar interface lebih sederhana, kadang opsi menu ada di "Other" sampai digunakan pertama kali.}*

Untuk *Web Service Name*, masukkan *DetermineRateService* , dan untuk *package*, masukkan *rate* Kemudian klik *Finish*.

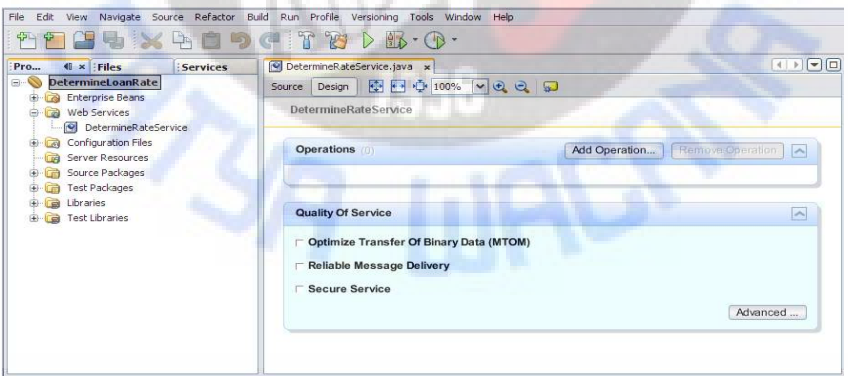
> *{Disini dibuat Web service baru dari awal, dan bukan dari EJB yang sudah ada.}*

Web service yang baru telah ditambahkan ke proyek dan view *Source* akan tampak :

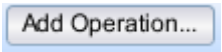


{Tanda @ merupakan anotasi. Ini mendukung pembuatan kode otomatis oleh Java. @WebService dan @Stateless menghasilkan pembuatan kode untuk stateless session EJBs dan Web service stubs,}

Pilih kontrol *Design* untuk berpindah dari mode Source ke mode Design. Kanvas *Design* akan tampak:



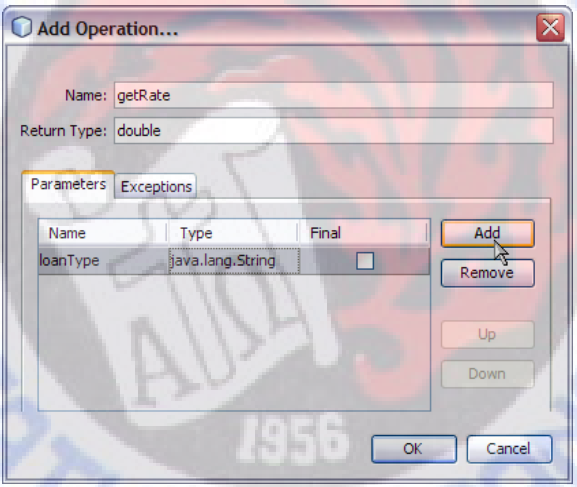
Klik pada tombol *Add Operation* untuk membuat sebuah operasi untuk implementasi web services. Windows wizard *Add Operation* akan muncul.



Untuk *Name*, ketik *getRate.* Untuk *Return Type*, ketik *double.*

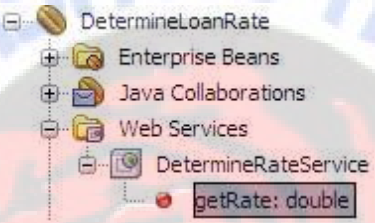
> *{Ketik "double" pada field Return Type. Tombol Browse dapat diguankan untuk memilih jenis objek, tetapi tidak membantu jika menggunakan tipe data primitif.}*

Pada tab *Parameters*, gunakan tombol *Add* untuk menetapkan argumen input dengan *Name* ketik *loanType* dan *Type* ketik java.lang.String (default):



Klik *OK* untuk mengakhiri pembuatan interface operasi *getRate*. Ingat bahwa operasi *getRate* terlihat pada view *Design*:

Ekspansi hirarki *Web Services* pada panel *Projects* untuk melihat bahwa operasi *getRate* telah ditambahkan.



Klik pada kontrol *Source* untuk berpindah ke view kode Java dan membuat implementasi dari metode *getRate* seperti gambar berikut ini :

```
17  @WebService()
18  @Stateless()
19  public class DetermineRateService {
20
21      /**
22       * Web service operation
23       */
24      @WebMethod(operationName = "getRate")
25      public double getRate(@WebParam(name = "loanType")
26      String loanType) {
27          if (loanType.equals("HOME")) return 7.0;
28  else if (loanType.equals("AUTO")) return 10.0;
29  else return 15.0;
30      }
31
32  }
```

Untuk memudahkan copy dan paste, berikut ini adalah kode untuk metode getRate():

```
if (loanType.equals("HOME")) return 7.0;
else if (loanType.equals("AUTO")) return 10.0;
else return 15.0;
```

Gunakan *Alt-Shift-F* sebagai short-cut untuk melakukan reformat kode. *Alt-Shift-F* merupakan short-cut untuk opsi menu *Source / Format*.

Save All seluruh pekerjaan.

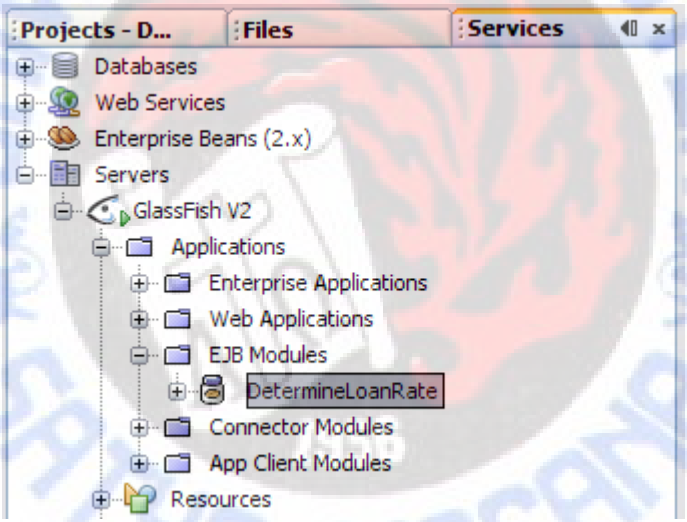
Lakukan *build* proyek, dengan melakukan klik kanan node *DetermineLoanRate* pada panel *Projects* dan pilih opsi *Clean and Build*. Jika tidak ada masalah maka akan terlihat pesan "*BUILD SUCCESSFUL*" pada panel *Output*.

{Jika panel Output tidak tampak,gunakan menu bar Window / Output / Output untuk menampilkannya.

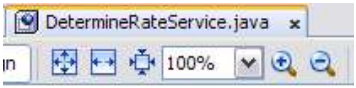
Perhatikan bahwa pesan juga tampak pada area status pada sisi kanan bawah dari window NetBeans.}

Lakukan deploy modul EJB ke server aplikasi GlassFish dengan melakukan klik-kanan pada node *DetermineLoanRate* pada panel *Projects* dan memilih opsi *Deploy*. Akan terlihat pesan "*BUILD SUCCESSFUL"* jika modul EJB telah berhasil dideploy.

Setelah deployment lengkap, cek untuk melihat apakah modul telah berhasil dideploy dengan melakukan klik pada tab *Services* dan melakukan ekspansi hirarki seperti gambar berikut ini:



{Jika modul tidak tampak, perlu dilakukkan klik-kanan pada EJB Modules dan pilih Refresh. Panel DetermineRateService.java dapat ditutup dengan klik kanan pada tab "X":}



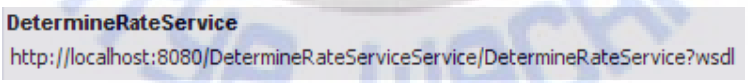
4.1.1 Eksplorasi WSDL dan XSD

WSDL (Web Services Description Language) merupakan dokumen XML yang mendeskripsikan interface untuk Web service. Jika service consumer mempunyai akses ke service provider, maka ia mempunyai akses untuk berkomunikasi dengan Web service.

Interface menyembunyikan rincian implementasi yang memungkinkan provider untuk misalnya merubah implementasi dari modul EJB ke aplikasi web, sepanjang interfacenya sama, klien tidak akan memperhatikan perbedaannya.

Untuk melakukan eksplorasi WSDL ini maka selanjutnya, lakukan klik pada tab *Projects* sehingga dapat terlihat hirarki proyek1.

Temukan *DetermineRateService* dibawah node *Web Services*. Lakukan klik-kanan dan pilih opsi *Properties* untuk mendisplay window *Properties*. Pada bagian bawah, tercatat bahwa WSDL telah dibuatkan dan tersedia pada sebuah URL:



Lakukan drag pada URL tersebut dan gunakan *Ctrl-C* untuk mengkopi ke clipboard. Klik pada tombol *Close*.

Pada web browser, lakukan, paste dari URL yang baru saja dikopi. Makan akan terlihat WSDL yang telahdibuat. Ini menjelaskan interface untuk service baru. Namun beberapa informasi kunci disimpan di suatu tempat dan ini hanya referensi saja.

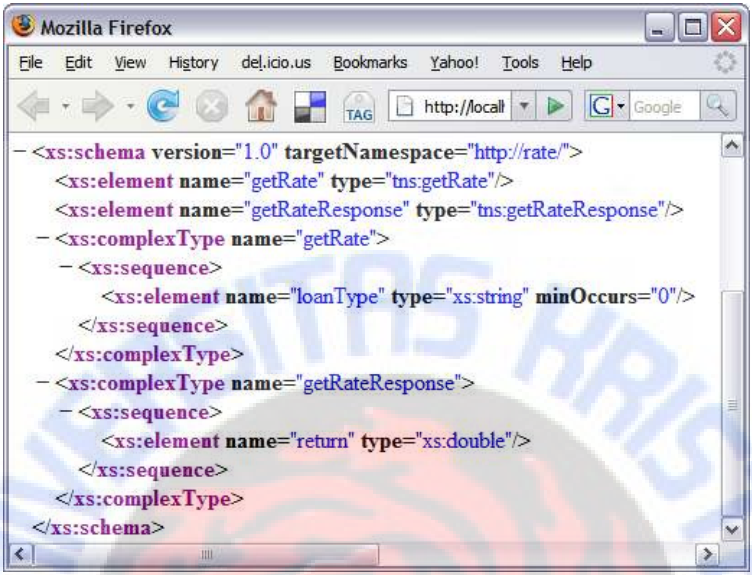
Source lainnya berupa dokumen skema XML direferensi pada bagian <*types*> dibawah ini:

```
- <definitions targetNamespace="http://rate/" name="DetermineRateServiceService">
    <wsp:UsingPolicy/>
  - <wsp:Policy wsu:Id="DetermineRateServicePortBinding_getRate_WSAT_Policy">
    - <wsp:ExactlyOne>
      - <wsp:All>
          <ns1:ATAlwaysCapability wsp:Optional="false"/>
          <ns2:ATAssertion ns3:Optional="true" wsp:Optional="true"/>
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://rate/" schemaLocation="http://localhost:8080/DetermineRateServiceService/DetermineRateService?xsd=1"/>
      </xsd:schema>
    </types>
```

Lakukan kopi URL dari bagian <*types*> ke dalam web browser baru untuk melihat struktur data input dan output yang terlibat dalam definisi interface web srvices. Ini adalah URL yang harus di-paste.

http://localhost:8080/DetermineRateServiceService/DetermineRateService?xsd=1

Maka akan terlihat XSD berikut ini.



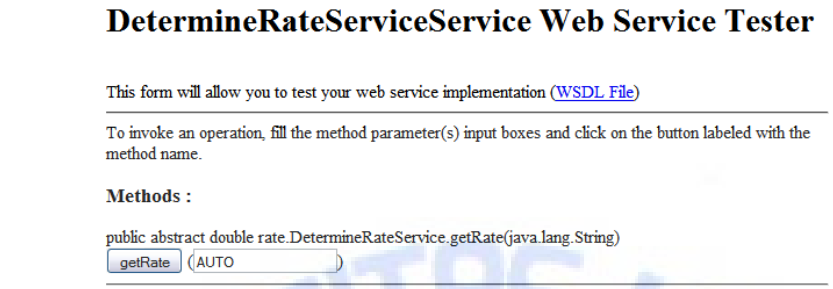
{Tampak rincian data definition yang menunjukkan Web service mengharapkan akan menerima struktur data getRate data yang berisi loanType string dan menjawab dengan struktur data getRateResponse yang berisi atribut yang dinamakan return dengan jenis data double.}

4.1.2 Testing Web Service

Akhirnya, lakukan pengujian web service. Klik kanan pada node *DeterminRateService* dan pilih *Test Web Service*. Web browser akan tampak.

{Perhatikan bahwa pada bagian atas halaman test, akan ditemukan link WSDL File yang menyediakan cara mudah untuk menampilkan WSDL.}

Ketikkan *AUTO* dan tekan tombol *getRate*:



{Pastikan melakukan klik pada tombol getRate dan bukan tombol Enter. Pastikan pula memasukkan HURUF BESAR.}

Web browser akan ditampilkan lagi dan terlihat SOAP request akan terkirim dan respon akan diterima. Berdasarkan pada kode Java dari method *getRate*, diharapkan bunga pinjaman adalah *10.0*:

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:getRateResponse xmlns:ns2="http://rate/">
            <return>10.0</return>
        </ns2:getRateResponse>
    </S:Body>
</S:Envelope>
```

Lakukan pengujian tambahan dengan jenis pinjaman selain RUMAH atau MIOBIL, misalnya KAPAL. Jika selain MOBIL atau RUMAH, maka nilai yang diharapkan adalah *15.0*.

4.2 Menghitung Pembayaran Pinjaman

Pada proyek yang lalu, telah dibuat modul EJB yang digunakan untuk menetapkan tingkat bunga pinjaman. Method getRate ditampilkan sebagai operasi web service. Pada proyek2 ini, akan dibangun fungsionalitas untuk menghitung pembayaran pinjaman. Sekedar variasi, maka penghitungan pembayaran pinjaman ini akan diimplementasikan sebagai endpoint web service berbasis servlet. Langkah ini sangat mirip dengan pendefinisian implementasi java untuk web services, dengan menyerahkan pada IDE Netbeans untuk menciptakan XSD, WSDL dan dokumen XML.

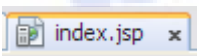
Pilih *File / New Project* dari *menu bar.* Wizard windows *New Project* akan tampak.

Pilih *Java Web* sebagai category dan *Web Application* untuk project type. Kemudian klik *Next*.

Untuk *Project Name*, ketik *CalculateLoanPayment* dan simpan dalam *Project Location*: *C:\latihanSOA*. Klik *Next*.

Terima semua default dengan klik *Finish* untuk mengakhiri dialog wizard. Proyek *CalculateLoanPayment* project tampak di panel *Projects*. Juga akan terlihat halaman *index.jsp*. Klik *Close*.

{Halaman index.jsp dapat ditutup jika tidak digunakan, dengan klik tab "X":



Sama seperti proyek1, pada proyek ini tidak digunakan cara yang benar, karena tidak terlebih dulu mendefiniskan interface (XSD dan WSDL). Untuk

pengujian cepat, maka akan menggunakan NetBeans dalam membuat interface secara otomatis.

Web service dapat dibuat lagi dengan EJB, namun agar variatif akan dibuat dengan aplikasi web.

Client Web service tidak dapat mengatakan perbedaan antara implementasi yang berbeda. Inilahh keindahan "information hiding" yang memisahkan interface dari implementasi.}

Pada node *CalculateLoanPayment*, klik-kanan dan pilih *New / Web Service*. Wizard window *New Web Service* akan tampak.

Untuk *Web Service Name*, ketik *CalculatePaymentService*

Untuk package, ketik *payment* dan klik *Finish*. Akan terlihat web service baru ditambahkan pada panel *Projects*.

Seperti yang telah dikerjakan di proyek1, klik pada ontrol *Design* untuk ke mode *Design* dan klik tombol *Add Operation* untuk membuat sebuah method baru. Wizard window *Add Operation* akan tampak.

Untuk *Name*, ketik: *getPayment*

Untuk *Return Type*, ketik: *double*

Pada tab *Parameters*, gunakan tombol *Add* tiga kali untuk mengetikkan argumen input:

- *interestRate* (*double*)
- *amount* (*double*)
- *period* (*int*)

{Gunakan box drop-down list pada kolom Type untuk memilih type data double dan int.}

Klik *OK* untuk mengakhiri pembuatan interface untuk operasi *getPayment*. Klik pada kontrol *Source* untuk pindah ke mode *Source*. Maka kerangka implementasi telah selesai dibuat.

Implementasikan method *getPayment* seperti gambar dibawah ini.

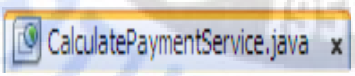
```
22      @WebMethod(operationName = "getPayment")
23      public double getPayment(@WebParam(name = "interestRate")
24      double interestRate, @WebParam(name = "amount")
25      double amount, @WebParam(name = "period")
26      int period) {
27          double rate = (interestRate / 12) / 100;
28  int months = period * 12;
29  return amount * (rate * Math.pow(1 + rate, months) / (Math.pow(1 + rate, months) - 1));
30      }
31
32  }
```

Berikut ini kode yang dapat dicopy-paste.

double rate = (interestRate / 12) / 100;
*int months = period * 12;*

*return amount * (rate * Math.pow(1 + rate, months) / (Math.pow(1 + rate, months) - 1));*

{Jika ruang editor kode Java akan diperluas, klik-double saja pada tab CalculatePaymentService.java dan panel akan diperluas:



Untuk kembali ke panel ukuran awal, klik-doble lagi ke tab.}

Gunakan shortcut *Alt-Shift-F* untuk penataan kode.

Klik *Save All.*

Lakukan *compile* Web service, dengan klik-kanan node *CalculateLoanPayment* dan memilih opsi *Clean and Build* option. Jika tidak ada error maka akan terlihat *"BUILD SUCCESSFUL."*

> *{Perhatikan window Output window untuk melihat feedback. Jika hasil Clean and Build ada pesan error message ini menunjukkan bahwa direktori tidak dapat dihapus, gunakan opsi Build saja (tidak dengan Clean).}*

Lakukan *deploy* modul web ke server aplikasi GlassFish dengan klik-kanan node *CalculateLoanPayment* dan memilih opsi *Deploy*. Akan terlihat pesan *"BUILD SUCCESSFUL"*.

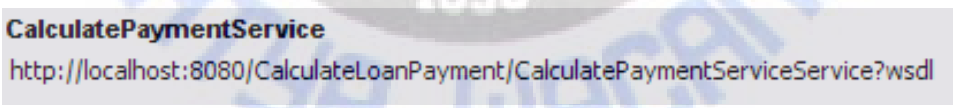
4.2.1 Eksplorasi WSDL dan XSD

Seperti yang telah dikerjakan di proyek1, akan dieksplorasi dokumen WSDL dan XSD yang secara otomatis dibuat oleh NetBeans.

Klik pada tab *Projects* dan temukan *CalculatePaymentService* dibawah node *Web Services* dalam proyek *CalculateLoanPayment*.

Klik-kanan dan pilih *Properties*.

Pada bagian bawah window, tercatat WSDL yang tersedia di URL:



Drag URL dan lakukan *Ctrl–C* untuk mengkopi ke clipboard. Pilih tombol *Close* pada window *Properties*.

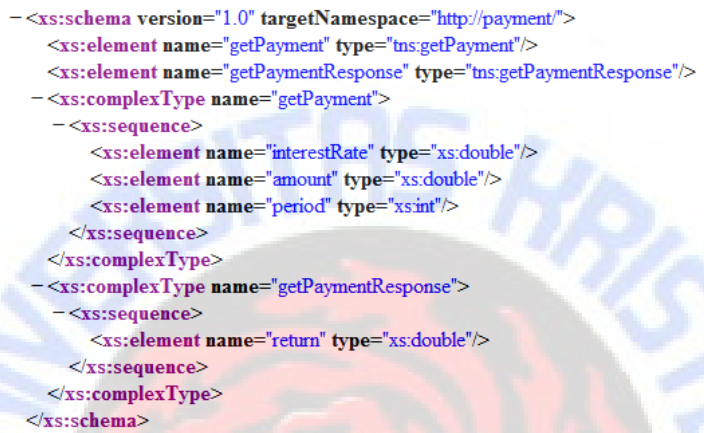
Pada web browser, lakukan paste URL. Akan terlihat WSDL yang telah dibuat. WSDL ini mendeskripsikan interface dari service.

```
−<definitions targetNamespace="http://payment/"
 name="CalculatePaymentServiceService">
 −<types>
  −<xsd:schema>
     <xsd:import namespace="http://payment/" schemaLocation="http://localhost:8080
     /CalculateLoanPayment/CalculatePaymentServiceService?xsd=1"/>
   </xsd:schema>
 </types>
 −<message name="getPayment">
    <part name="parameters" element="tns:getPayment"/>
 </message>
 −<message name="getPaymentResponse">
    <part name="parameters" element="tns:getPaymentResponse"/>
 </message>
 −<portType name="CalculatePaymentService">
  −<operation name="getPayment">
     <input message="tns:getPayment"/>
     <output message="tns:getPaymentResponse"/>
   </operation>
 </portType>
 −<binding name="CalculatePaymentServicePortBinding"
 type="tns:CalculatePaymentService">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  −<operation name="getPayment">
     <soap:operation soapAction=""/>
    −<input>
       <soap:body use="literal"/>
     </input>
    −<output>
       <soap:body use="literal"/>
     </output>
   </operation>
 </binding>
```

{Disini ada dua jenis pesan yang dibuat berdasarkan dua elemen pada XSD. Port Type menyediakan definisi interface yang menunjukkan bahwa operasi getPayment tersedia, yang mengharapkan pesan input dan mengirimkan pesan output berdasarkan jenis pesan diatas. Untuk bagian konkrit dari WSDL, kaitkan operasi ke SOAP sebagai sebuah protocol.}

Seperti yang telah dikerjakan pada proyek terakhir, temukan baris yang melakukan import skema XML untuk mendefinisikan pesan input dan output:

Lakukan kopi URL ini ke web browser lain untuk melihat definisi jenis data web service sisanya.



{Copy: http://localhost:8080/CalculateLoanPayment/ CalculatePay mentServiceService?xsd=1

Ada dua complex types yang didefinisikan. Dan ini akan menjadi dasar untuk dua elemen. Complex type pertama mendefinisikan inputs (interestRate, amount dan period). Yang kedua mendefinisikan nilai return dengan tipe double.}

4.2.2 Testing Web Service

Lakukan pengujian Web service. Klik-kanan pada node *CalculatePaymentService* pada panel *Projects* dan pilih opsi *Test Web Service*. Web browser akan tampak.

Ketikkan input berikut :

- *interestRate*: *10.0*
- *amount*: *100000.00*
- *period*: *15*

{ Ini sama seperti yang dipelajari di proyek diatas Implementasi web service dibuat dan membiarkan Netbeans membuat XSD dan dokumen WSDL secara otomatis.}

CalculatePaymentServiceService Web Service Tester

This form will allow you to test your web service implementation (WSDL File)

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract double payment.CalculatePaymentService.getPayment(double,double,int)

getPayment (10.0 , 100000.00 , 15)

Klik tombol *getPayment*.

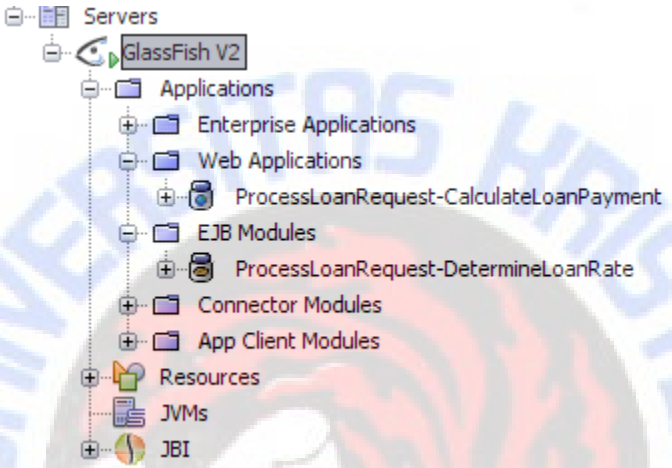
{Terlihat bahwa data type adalah double, double dan int. Lihat acuan pada type complex getPayment dallam XSD untuk melihat urutan parameter.}

Akan terlihat SOAP Response:

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:getPaymentResponse xmlns:ns2="http://payment/">
            <return>1074.6051177081179</return>
        </ns2:getPaymentResponse>
    </S:Body>
</S:Envelope>
```

Klik pada tab *Services* sehingga terlihat aplikasi web (*CalculateLoanPayment*) yang telah berhasil dideploy:



{Perhatikan hirarki pada node JBI. Sebelum dikerjakan, maka aplikasi komposit akan dideploy dengan framewor JBI pada server Glassfish. }

Klik Close panel *CalculatePaymentService.java* dan panel editor lain sebelum pindah ke proyek lain dimana akan dibuat proses BPEL yang digunakan untuk orkestrasi ke dua web services tersebut.

{Cara cepat menutup semua panel tab adalah dengan klik-kanan pada salah satu tab dan pilih opsi Close All Documents.}