Bab 8
Web Services

8.1 Tujuan

Diakhir pembahasan, diharapkan siswa dapat :

- Mengetahui bagaimana memodelkan data menggunakan XML
- Mengetahui beberapa Java API yang digunakan dalam pemrosesan XML
- Mendeskripsikan apa yang bisa dilaksanakan oleh web service
- Mengetahui beberapa *key standard* dalam web service
- Mengetahui bagaiamana membuat web service mobile sebagai client

8.2 Pengenalan terhadap XML

XML (eXtensible Markup Language) adalah sebuah *meta-language* untuk mendeskripsikan data. XML merupakan sebuah cara merepresentasikan data tanpa tergantung kepada system. Ia juga dapat digunakan sebagai *extension markup languages*. XML adalah berbasis text, sehingga ia dapat dengan mudah dipindahkan dari satu sistem komputer ke sistem yang lain. Dengan XML, data direpresentasikan dalam sebuah dokumen yang terstruktur dan ia juga telah menjadi sebuah standard industri.

Element

Sebuah dokumen XML adalah sebuah dokumen yang mudah dibaca dan terdiri dari XML tag atau element. Sama halnya dengan HTML, XML tag didefinisikan dengan kurung siku <>. Sebuah dokumen XML memiliki struktur seperti entities didalam sebuah tree.

Anda dapat menggunakan tag sesuai dengan yang Anda inginkan, selama semua aplikasi yang menggunakan dokumen tersebut menggunakan tag dengan nama yang sama. Tag dapat memiliki attributes. Dalam contoh dibawah ini, "task" pertama memiliki "id" dengan attribut "1", sedangkan "task" yang kedua memiliki "id" dengan attribute "2".

```
<tasks>
  <task id="1">
```

```
    <name>connect to database</name>
    <duration>2</duration>
    <assignedTo>alex</assignedTo>
    <progress>50</progress>
  </task>
  <task id="2">
    <name>list table rows</name>
    <duration>4</duration>
    <assignedTo>alex</assignedTo>
    <progress>100</progress>
  </task>
</tasks>
```

Attributes

Tag dapat juga terdiri dari attribute-attribute. Didalam contoh, tag "task" memiliki attribute dengan nama "id". Sebuah attribut diikuti dengan tanda sama dengan (=) dan diikuti dengan value atau nilainya.

Pada saat mendesai sebuah struktur XML, permasalahan yang selalu muncul adalah apakah sebuah data element harus menjadi attribute dari sebuah element atau menjadi sebuah sub-element. Seperti contoh, XML diatas juga dapat kita tulis lagi sebagai berikut:

```
<tasks>
  <task>
     <id>1</id>
     <name>connect to database</name>
   ...
</tasks>
```

Tidak ada aturan yang pasti, struktur element seperti apa yang harus kita anut. Akan tetapi dalam beberapa situasi, aturan-aturan dibawah ini harus dipenuhi:

- Data akan memiliki beberap sub-struktur, pada kasus dimana Anda harus menggunakan sebuah sub-element karena ia tidak boleh dimodelkan sebagai attribut.
- Data akan terdiri dari beberapa baris apabila attribut ingin dibuat sesederhana mungkin – sebuah string yang pendek tetapi mudah untuk dibaca dan digunakan.

- Data element dimungkinkan untuk muncul kembali.
- Data akan sering berubah.

XML Schema

XML tag harus bersifat extensible, dimana seorang desainer system dimungkinkan untuk menuliskan sendiri XML tag-nya dalam pendeskripsian sebuah content. Anda dapat menciptakan tag-tag yang berbeda untuk setiap format dokumen yang Anda inginkan didalam aplikasi atau sistem Anda.

Tag didefinisikan menggunakan XML schema language. Sebuah schema mendefinisikan struktur dari dokumen XML. Sebuah skema juga digunakan membatasi content dari sebuah dokumen XML kedalam sebuah element, attributes, dan values tertentu.

Sebuah Document Type Definition schema adalah bagian dari spesifikasi XML. Kita akan memanggil schema yang ditulis dalam bahasa ini disebut sebagai DTD. DTD ini juga mendefinisikan tag atau attribute mana yang sangat diperlukan dan mana yang bersifat optional.

```
<!ELEMENT tasks (task)*>
<!ELEMENT task (name, duration, assignedTo, progress) >
<!ATTLIST task
  id CDATA #REQUIRED
>
<!ELEMENT task (name, duration, assignedTo, progress) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT duration (#PCDATA) >
<!ELEMENT assignedTo (#PCDATA) >
<!ELEMENT progress (#PCDATA) >
```

Namespace

Ada beberapa kasus dimana tag atau element Anda memiliki nama yang sama. Misalnya, ada beberapa element yang mempunya nama yang sama yaitu “name”. Untuk mengatasi hal tersebut, sebuah namespace digunakan. Spesifikasi dari namespace akan digunakan oleh penulis dokumentasi untuk mengetahui schema atau DTD mana yang digunakan pada element tertentu. Namespace dapat diaplikasikan pada attribut dan juga pada elements.

8.3 Java APIs bagi XML

JAXP

JAXP atau Java API for XML Processing (JAXP) adalah sebuah fleksible API yang mendukung Anda untuk mendukung sembarang XML-complient parser didalam aplikasi Java Anda. Ia memiliki sebuah layer plugability, dimana Anda dapat menambahkan sembarang implementasi dari SAX atau DOM APIs.

JAXP juga telah mendukung namespaces, sehingga ia akan mendukung Anda untuk bekerja dengan schema XML dalam mengatasi permasalahan penamaan.

DOM API

Document Object Model (DOM) adalah sebuah struktur tree, dimana tiap node akan terdiri dari sebuah komponen dari struktur XML. DOM API mendukung Anda untuk membuat atau menghilangkan sebuah node didalam tree. Ia juga dapat digunakan untuk mengganti content dari node dan mengubah hierarki node.

Oleh karena, sebuah tree me-representasikan keseluruhan XML data, DOM API membutuhkan banyak memori didalam runtime-nya.

SAX API

Simple API for XML (SAX) adalah sebuah event-based XML parser API. Ia akan membaca XML dokumen dari awal hingga akhir. Setiap saat bertemu dengan sebuah construction syntax, ia akan memberikan tanda (notify) untuk menjalankan program tersebut. Sebuah SAX parser akan memberikan tanda kepada aplikasi dengan jalan memanggil method dari interface ContentHandler.

Sebagai contoh, pada saat parser mencapai simbol kurang dari (<), ia akan memanggil method startElement. Pada saat bertemu dengan tag terakhir (sebuah slash yang diikuti dengan tanda lebih besar dari), hal ini disebut method endElement.

SAX sangatlah cepat dan efisien. Ia membutuhkan memori yang lebih sedikit daripada DOM, karena ia tidak mengharuskan untuk membuat sebuah struktur tree internal dari data-data XML. SAX hanya akan mengirimkan data setiap kali aplikasi ingin membaca data tersebut.

XLST API

Extensible Stylesheet Language Transformation (XSLT) standard mendefinisikan mekanisme untuk pengalamatan (addressing) data-data XML dan untuk menspesifikkan transformasi data.

Extensible Stylesheet Language (XSL) memiliki tiga sub-komponen:

XSL-FO – Formatting Objects standard. XSL-FO adalah sebuah standard yang menyediakan mekanisme untuk mendeskripsikan aspek-aspek dari sebuah object misalnya ukuran huruf dan layout halaman. Sub komponen ini tidak tercover didalam JAXP.

XSLT – adalah sebuah transformation language dimana Anda diharapkan dapat mendefisinikan sendiri transformasi dari sebuah XML ke format yang lain seperti HTML.

Xpath – adalah sebuah language spesification, dimana Anda diharapkan dapat menspesifikasikan sendiri bagian-bagian dari struktur XML yang direference setiap saat. Xpath adalah sebuah mekanisme pengalamatan yang mendukung Anda untuk mendefinisikan sebuah path kepada element.

JAX-RPC

Java API for XML-based RPC (JAX-RPC) adalah sebuah teknologi untuk membangun web services dan client web service yang dapat mengaktifkan remote procedure calls (RPC). RPC model mendukung client untuk mengeksekusi procedure pada remote system.

Didalam JAX-RPC, remote procedure calls dan hasilnya direpresentasikan pada protocol berbasis XML. SOAP adalah salah satu protokol yang menggunakan JAX-RPC. JAX-RPC akan meng-convert API pemanggilan dan beraksi dari dan untuk SOAP/XML messages. Hal ini akan menyebabkan SOAP menjadi lebih mudah dan menghilangkan kekompleksan.

8.4 Web Services

Web Services Messaging

W3C mendefinisikan web service sebgai "sebuah software aplikasi yang dapat teridentifikasi oleh URI dan memiliki interface yang didefiniskan, dideskripsikan, dan dimengerti oleh XML dan juga mendukung interaksi langsung dengan software aplikasi yang lain dengan menggunakan message berbasis XML melalui protokol internet"

Web service adalah sebuah sofware aplikasi yang tidak terpengaruh oleh platform, ia akan menyediakan method-method yang dapat diakses oleh network. Ia juga akan menggunakan XML untuk pertukaran data, khususnya pada dua entities bisnis yang berbeda.

Beberapa karakteristik dari web service adalah:

- Message-based
- Standards-based
- Programming language independent
- Platform-neutral

Beberapa key standard didalam web service adalah: XML, SOAP, WSDL and UDDI.

SOAP (Simple Object Access Protocol) adalah sebuah XML-based mark-up language untuk pergantian pesan diantara aplikasi-aplikasi. SOAP berguna seperti sebuah amplop yang digunakan untuk pertukaran data object didalam network. SOAP mendefinisikan empat aspek didalam komunikasi: Message envelope, Encoding, RPC call convention, dan bagaimana menyatukan sebuah message didalam protokol transport.

Sebuah SOAP message terdiri dari SOAP Envelop dan bisa terdiri dari attachments atau tidak memiliki attachment. SOAP envelop tersusun dari SOAP header dan SOAP body, sedangkan SOAP attachment membolehkan non-XML data untuk dimasukkan kedalam SOAP message, di-encoded, dan diletakkan kedalam SOAP message dengan menggunakan MIME-multipart.

Web Services Description

WSDL (Web Services Description Language) adalah sebuah XML-based language untuk mendeskripsikan XML. Ia menyediakan service yang mendeskripsikan service request dengan menggunakan protokol-protokol yang berbeda dan juga encoding. Ia akan memfasilitasi komunikasi antar aplikasi. WSDL akan mendeskripsikan apa yang akan dilakukan oleh web service, bagaimana menemukannya dan bagaimana untuk mengoperasikannya.

Spesifikasi WSDL mendefinisikan tujuh tipe element:

- Types – element untuk mendefinisikan tipe data. Mereka akan mendefinisikan tipe data (seperti string atau integer) dari element didalam sebuah message.
- Message - abstract, pendefinisian tipe data yang akan dikomunikasikan.
- Operation – sebuah deskripsi abstract dari sebuah action yang didukung oleh service.

- Port Type – sebuah koleksi abstract dari operations yang didukung oleh lebih dari satu endpoints.
- Binding – mendefinisikan penyatuan dari tipe port (koleksi dari operasi-operasi) menjadi sebuah protokol transport dan data format (ex. SOAP 1.1 pada HTTP). Ini adalah sebuah protokol konkret dan sebuah spesifikasi data format didalam tipe port tertentu.
- Port – mendefinisikan sebuah komunikasi endpoint sebagai kombinasi dari binding dan alamat network. Bagi protokol HTTP, ini adalah sebuah bentuk dari URL sedangkan bagi protokol SMTP, ini adalah sebuah form dari email address.
- Service – satu set port yang terkorelasi atau suatu endpoints.

WSDL mendefinisikan service sebagai sebuah koleksi dari endpoints network. Sebuah definisi abstrak dari endpoints dan messages adalah ia bersifat terpisah dari pembangunan network atau penyatuan data format. Pembagian ini menyebabkan penggunaan kembali abstract description dari data yang akan dipertukarkan (message exchange) dan abstract collection dari operasi (ports)

Protokol konkret dan spesfikasi data format bagi tipe port tertentu menentukan binding yang dapat digunakan kembali(reusable). Sebuah port adalah sebuah network address yang dikombinasikan reusable binding; sebuah service adalah koleksi dari port-port.

Web Service Discovery

UDDI (Universal Description, Discovery and Integration) adalah sebuah service registry bagi pengalokasian web service. UDDI mengkombinasikan SOAP dan WSDL untuk pembentukan sebuah registry API bagi pendaftaran dan pengenalan service. Ia menyediakan sebuah area umum dimana sebuah organisasi dapat mengiklankan keberadaan mereka dan service yang mereka berikan (web service).

UDDI adalah sebuah framework yang mendefinisikan sebuah XML-based registry dimana sebuah organisasi dapat meng-upload informasi mengenai service yang mereka berikan. XML-based registry berisi nama-nama dari organisasi tsb, beserta service dan deskripsi dari service yang mereka berikan.

8.5 J2ME Web Services API (WSA)

Web Services API(JSR 172) menyediakan fungsi-fungsi tambahan yang mendukung web service. API ini berisi fungsi-fungsi dasar yang digunakan dalam web service client seperti remote web invocation dan XML parsing. WSA hanya merupakan subset dari J2SE API.

WSA hanya mendukung pemakaian dari web service. Hal ini berarti, aplikasi J2ME dengan menggunakan WSA hanya dapat menjadi konsumen dari service dan bukan merupakan producer service. UDDI juga tidak disupport oleh JSR 172.

JAXP subset disupport oleh WSA spesification yang didukung oleh SAX. Ia tidak berisi dukungan bagi DOM dan XSLT.

Dokumen parsing XML menggunakan SAX

Untuk mendapatkan instance dari SAX Parser, pertama-tama kita harus mendapatkan instance dari SAXParserFactory:

```
// Dapatkan instance dari SAX parser factory
SAXParserFactory factory = SAXParserFactory.newInstance();
```

Kemudian, kita akan mendapatkan instance dari SAX Parser:

```
SAXParser parser = factory.newSAXParser();
```

Pada akhirnya, kita akan membuat sebuah source input dan menggunakan event handler untuk mem-parser. Untuk mempermudah contoh, kita akan membuat sebuah stream dari sebuah String. Biasanya, hal ini bisa terlaksana dengan menggunakan resource dari file atau network. Contoh yang kita buat juga tidak memiliki sebuah GUI-ia hanya mencetak sebuah outcome dari sebuah parsing menjadi sebuah standard output.

```
ByteArrayInputStream stream =
    new ByteArrayInputStream(sampleXML.getBytes());
InputSource inputSource = new InputSource(stream);
SaxEventHandler handler = new SaxEventHandler();
parser.parse(inputSource, handler);
```

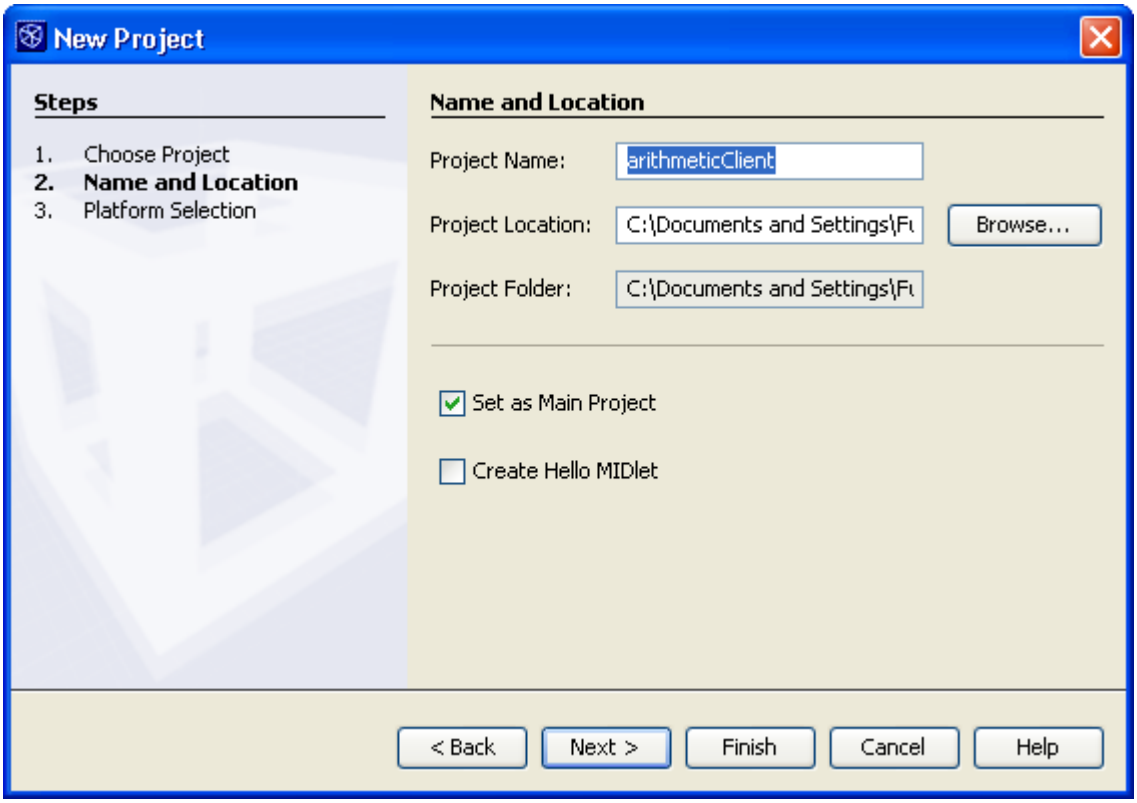
Kode berikut ini adalah kode untuk Event Handler:

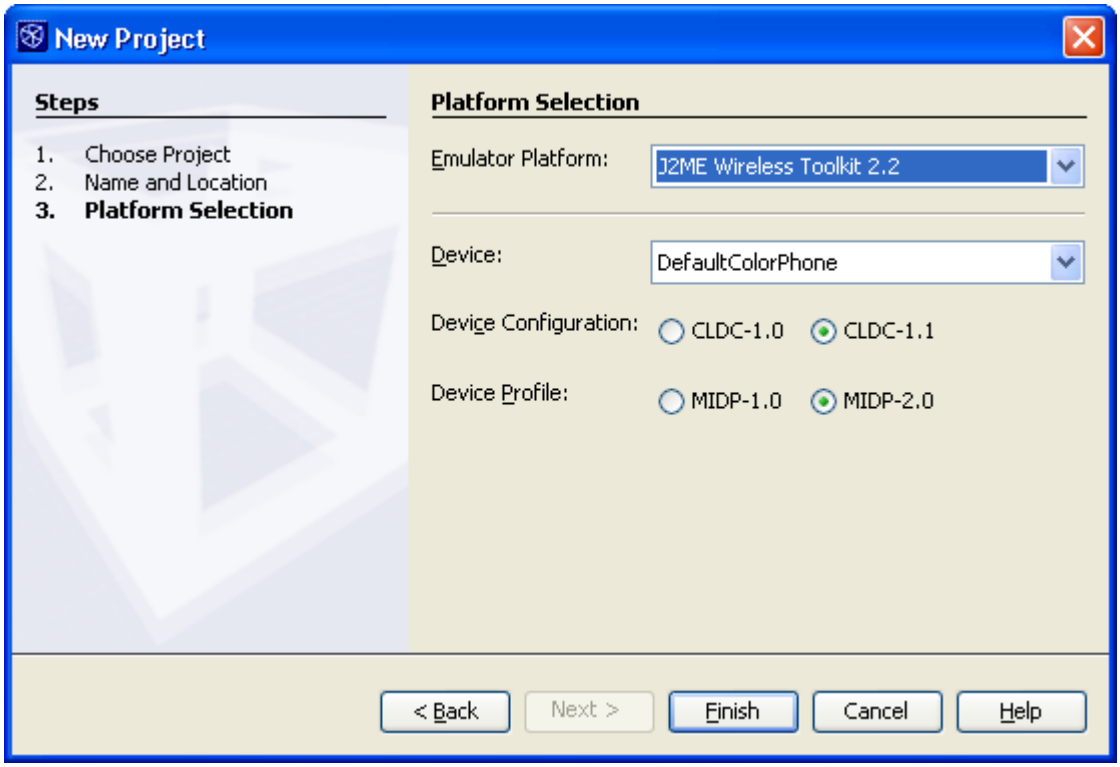
```
import java.util.*;
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;

public class SaxEventHandler extends DefaultHandler {
    private boolean finished;
    private Stack qNameStack = new Stack();
    private Vector tasks = new Vector();

    private static final String TASKS_ELEMENT = "tasks";
    private static final String TASK_ELEMENT = "task";
    private static final String NAME_ELEMENT = "name";
    private static final String ID_ATTRIBUTE= "id";

    public void startDocument() throws SAXException {
        finished = false;
        qNameStack.removeAllElements();
        tasks.removeAllElements();
    }

  public void endDocument()
          throws SAXException {
      finished = true;

      // Akhir dari dokumen, sekarang lakukan proses untuk memparsing object
      for (int i=0; i<tasks.size(); i++) {
          Task task = (Task) tasks.elementAt(i);
          System.out.println("Task: " + task.name);
      }
  }

  public void startElement(
          String uri,
          String localName,
          String qName,
          Attributes attributes)
          throws SAXException {

      if (TASK_ELEMENT.equals(qName)) {
            // Dapatkan id attribute dari sebuah task
          String id = attributes.getValue(ID_ATTRIBUTE);
```

```
            Task task = new Task(id);
            tasks.addElement(task);
        }

        qNameStack.push(qName);
    }

    public void characters(
            char[] ch,
            int start,
            int length)
            throws SAXException {

        String name = new String(ch, start, length);

        String qName = (String) qNameStack.peek();
        if (NAME_ELEMENT.equals(qName)) {
            if (tasks.size() > 0) {
                Task task = (Task) tasks.lastElement();
                task.name = name;
            }
        }
    }

    public void endElement(
            String uri,
            String localName,
            String qName)
            throws SAXException {

        qNameStack.pop();
    }
}
```

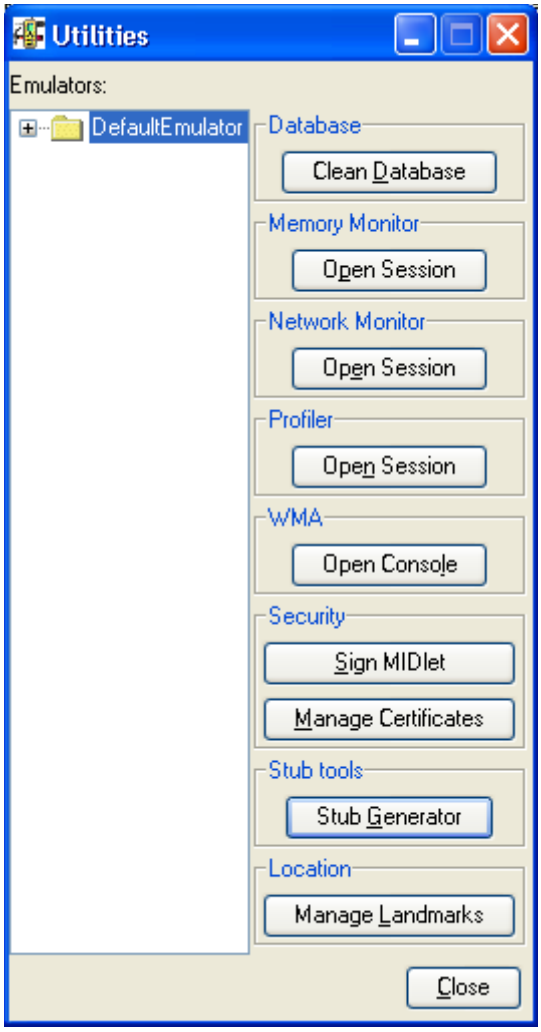
8.6 Membuat sebuah Mobile Web Service Client

Kita akan membuat secara sederhana aplikasi mobile dan kita akan menamakannya "arithmeticClient".





Kemudian, kita akan menggunakan stub generator (Tools->Java Platform Manager -> Wireless Toolkit -> Tools and Extensions -> Open Utilities -> Stub Generator):

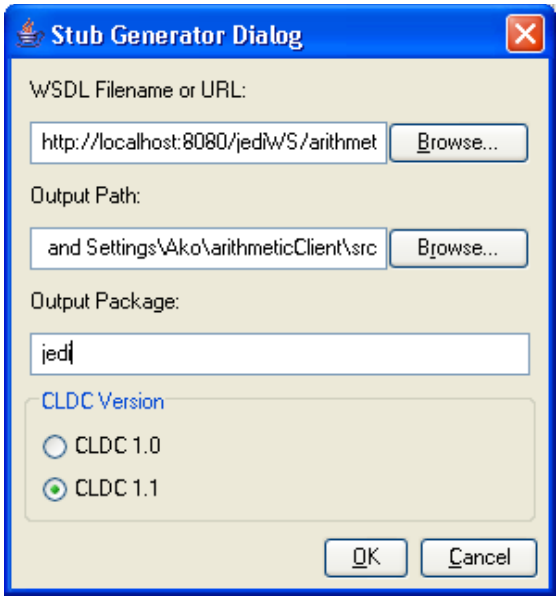


Sebelum kita dapat membuat sebuah web service client, kita harus memiliki sebuah file WSDL atau lokasi URL dimana terdapat service yang dapat kita pergunakan. Seperti yang telah disebutkan dalam bagian sebelumnya, JSR 172 tidak mendukung UDDI atau automatic discovery bagi service.

Masukkan lokasi WSDL dari sebuah web service. Path dari outputnya harus menjadi path dari direktori sumber project Anda (PROJECT_PATH/src). Sebuah stub generator tidak menerima package tanpa penamaan. Oleh karena itu, Anda harus meng-inputkan sebuah nama package.

Kemudian, kita akan membuat sebuah Midlet yang akan menggunakan stub yang telah kita buat untuk mengakses sebuah web service:

```
/*
 * arithemeticClient.java
 *
 */

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import jedi.*;

public class WSClient extends MIDlet {
    private ArithmeticSEI_Stub stub;

    public void startApp() {
        System.out.println("Creating stub...");
        stub = new ArithmeticSEI_Stub();
```




```
        System.out.println("Invoking operation...");
        WebServiceClient client = new WebServiceClient();
        Thread thread = new Thread(client);
        thread.start();
```

```
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }


    class WebServiceClient implements Runnable {
        public void run() {
            try {
                int reply = stub.multiply(4, 5);

                System.out.println("Reply: " + reply);

            } catch (java.rmi.RemoteException rex) {
                System.out.println("Remote Exception: " + rex.getMessage());
            }
        }
    }
}
```