# JSR 172: Accessing a .NET Web Service

This article demonstrates how the JSR-172: J2ME™ Web Services JAX-RPC package can be used to consume a .NET web service.

For this example, we're going to access this .NET web service:

```
http://www.webservicex.net/stockquote.asmx
```

If you navigate there, you'll see a description of the service, and it's one method: GetQuote(String symbol). We're going to invoke that method remotely from a Java ME MIDlet.

## Generating the Stub Class

We need the Web Service Description Language (WSDL) description for the service. Luckily, .NET services supply this for us if we append "?WSDL" to the URL.

The Java ME SDK contains a tool called "wscompile", that reads the WSDL data and generates a "stub" Java class. This class acts as a local proxy for the remote service. You call a method in the stub class, and it calls the remote method for you.

**Tip:** The Nokia SDK does not include the wscompile tool. If you are using the Nokia SDK and want to generate the stubs to simplify the connection to a web service, you need to download and install the Oracle's Java ME SDK, which can be downloaded here. After installing it, you will find wscompile (among other tools) in the bin directory.

To generate the stub class, we need a config.xml file.

```
<?xml version="1.0"?>
<configuration xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
 <wsdl location="http://www.webservicex.net/stockquote.asmx?WSDL" packageName="rpcdemo" />
</configuration>
```

The wsdl location must match the URL for the service (with "?WSDL" tacked on the end). The packageName is the package for the generated files.

```
\Java_ME_platform_SDK_3.0\bin\wscompile.exe -gen -cldc1.1 config.xml
```

You should specify -cldc1.1 if the web service might use floats or doubles as arguments or return value.

After executing this, you should have a file (amongst others) for the class: rpcdemo.StockQuoteSoap_Stub

Add the generated classes to your project in your IDE (for this example, you only need to add these two classes: StockQuoteSoap and StockQuoteSoap_Stub).

## Using the Stub Class

Here's a MIDlet that will invoke the GetQuote method (which wscompile has renamed getQuote(), to match the Java convention).

Since the method is remote, invoking it might be time consuming, so it needs to be done in a separate thread. Therefore, the interesting part of the following code is the run() method.

```
package rpcdemo;

import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;

public class RpcDemo extends MIDlet implements CommandListener, Runnable {

    private Form form;

    public void startApp() {
        if (form == null) {
            form = new Form("RpcDemo");
            form.addCommand(new Command("Exit", Command.EXIT, 0));
            form.setCommandListener(this);

            // get the data
            (new Thread(this)).start();
        }
        Display.getDisplay(this).setCurrent(form);
    }

    public void pauseApp() {
        // empty
    }

    public void destroyApp(boolean must) {
        // empty
    }

    public void commandAction(Command c, Displayable d) {
        if (c.getCommandType() == Command.EXIT) {
```


(C) Copyright Nokia 2013. All rights reserved.

```
            notifyDestroyed();
        }
    }

    public void run() {
        try {
            // create the stub
            StockQuoteSoap_Stub service = new StockQuoteSoap_Stub();
            // set the URL for the service
            service._setProperty(javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY, "http://www.webservicex.net/stockquote.asmx");

            println("Connecting...");
            // invoke the remote method
            String xmlResponse = service.getQuote("NOK");
            println(xmlResponse);

            println("Done.");
        } catch (Exception e) {
            println(e.toString());
        }
    }

    private void println(String s) {
        form.append(s + "\n");
    }
}
```

## Processing the Response

The response comes back as a String, which should look something like:

```
<StockQuotes>
 <Stock>
  <Symbol>NOK</Symbol>
  <Last>12.76</Last>
  <Date>4/23/2010</Date>
  <Time>4:00pm</Time>
  <Change>-0.23</Change>
  <Open>12.70</Open>
  <High>12.76</High>
  <Low>12.57</Low>
  <Volume>50259424</Volume>
  <MktCap>47.317B</MktCap>
  <PreviousClose>12.99</PreviousClose>
  <PercentageChange>-1.77%</PercentageChange>
  <AnnRange>12.10 - 16.58</AnnRange>
  <Earns>0.00</Earns>
  <P-E>N/A</P-E>
  <Name>Nokia Corporation</Name>
 </Stock>
</StockQuotes>
```

Since this is XML, you can process it using the XML parsing features of JSR 172. For more information about that, see JSR 172: XML Parsing Example.

## Source Code

A zip file containing the example application's project can be downloaded here: NetWebService Test.zip


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.