Building Web Applications with NetBeans IDE and SAP Business One Web Services

Applies to:
SAP Business One 2007, Business One Web Services, Business One DI Server

Summary
This article demonstrates how developers can extend SAP Business One functionality using SAP Business One Web Services and Sun Microsystems’ NetBeans IDE. The article details out how to use the NetBeans visual composer to rapidly create Web applications, drag and drop Web Services operations into the IDE editor, and enable tracing of the SOAP messages with the Message Handler module.

Author: Marina Fisher
Company: Sun Microsystems
Created on: 04 February 2008

Author Bio
Marina Fisher is an enterprise architect working at Sun Microsystems. She specializes in Web 2.0, SOA, Java EE, as well as Sun and SAP technologies. She is an author of the book Java-.NET Interoperability and a Sun Certified Enterprise Architect for Java EE.
# Table of Contents

Creating Web Applications with NetBeans IDE and SAP Business One Web Services .............................................3
  Installing SAP Business One Web Services Plug-In .................................................................................................4
  Creating New NetBeans Project ...............................................................................................................................7
  Creating Business One Web Service Clients ...........................................................................................................12
  Working with Business One WSDLs ........................................................................................................................16
  WSDL Enumeration Types ........................................................................................................................................18
  Passing Session ID Within the User Session ...............................................................................................................20
  Tracing DI Server SOAP Messages with Java Message Handler .............................................................................21
  Java Message Handler ...............................................................................................................................................22
  SAOPFault Processing ...............................................................................................................................................28

Conclusion .................................................................................................................................................................33

Appendix A .................................................................................................................................................................34
  Login Service ..........................................................................................................................................................34

Appendix B .................................................................................................................................................................35
  Add a business partner with addresses via BusinessPartnersService .................................................................35

Appendix C .................................................................................................................................................................36
  Update the BP with BusinessPartnersService .......................................................................................................36

Appendix D .................................................................................................................................................................38
  Add a A/R Invoice via InvoiceService ..................................................................................................................38

Related Content ..........................................................................................................................................................39

Disclaimer and Liability Notice ..................................................................................................................................40
Creating Web Applications with NetBeans IDE and SAP Business One Web Services

This tutorial outlines basic steps on how to develop a visual Web application using open source NetBeans IDE, the NetBeans Plugin for SAP Business One Web Services, and SAP Business One Web Services. The NetBeans IDE is an Integrated Development Environment to create Web, desktop, enterprise, and mobile applications. The IDE supports Java, C/C++, PHP, and Ruby. The IDE visual editor allows rapidly develop GUI using Ajax, CSS, JavaScript, and JSF. The visual editor also simplifies visual components integration with databases and ability to manipulate with the backend data. The IDE allows linking web pages of your application using the Visual Page Flow Editor, supporting JSF, JSP, and HTML pages. NetBeans runs on many platforms including Windows, Linux, Mac OS X and Solaris. The IDE is easy to install and use straight out of the box.

The purpose of the NetBeans Plugin for SAP Business One Web Services is to provide a single step for creating references for all of the Business One Web Services within your project. There are a few components necessary to run the Plug In, they include Java runtime, NetBeans IDE, Glassfish application server, and the Plug In.

To start with, you should have a Java runtime environment and the latest version of NetBeans installed on your development environment. When downloading the IDE, select ‘Java download’ column to enable Java Web features. The Glassfish application server has been used for testing the Plug In, and the application server comes as part of the IDE download. Tomcat web server has not been tested but is also supported with the IDE and is downloaded as part of the same bundle. After setting up the IDE, you can create a web application and import Business One Web Services by following steps below.
Installing SAP Business One Web Services Plug-In

Open the IDE and under Tools menu select Plugins, Tools->Plugins. Scroll down to locate and select SAP Business One Web Services, as captured below:
Click Install button at the bottom of the popup window. It will prompt you to proceed with the installation, select Agree on the License Agreement page and then select continue on the Validation Warning screen.
Once the installation is complete, you should see the following message:

![NetBeans IDE Installer](image)

Click Finish and close the Plugins popup window.

Alternatively, you can download the plugin locally to your machine from:

http://plugins.netbeans.org/PluginPortal/faces/PluginDetailPage.jsp?pluginid=8116

From within the IDE, open to Tools ->Plugins menu. Select Downloaded tab and navigate to the selected plugin. You should see SAP Business One Web Servicews plugin selected. Click on 'Install' button at the bottom of the popup window to install the plugin. Navigate to 'Installed' tab and you should see the plugin listed.

Now you are ready to create a project and using the installed Plugin import all of the SAP Business One Web Services into the project to build a web application.
Creating New NetBeans Project

Open the IDE and create a new project by selecting File -> New Project
Select a project type, say Web application, and populate project name and location:
Select Visual Web JavaServer Faces framework to assemble an application with ready to use palette of visual components. Specify a package name for your application.

Press Finish.
Once the project is created and opened, select a visual Palette on the right hand side to drag and drop widgets onto the design panel.
You can change properties of individual widgets in the Properties area, located in the lower right hand corner, when the Design View of the page is selected. Here you can select the widget and change its properties, for example rename the button to be ‘login’ button.
Creating Business One Web Service Clients

Now you are ready to import the Business One DI Server Web Services into the IDE using the pre-installed Plugin. The IDE will automatically generate the Web Service references.

Select your project, and right mouse click on it, at the bottom menu select Add SAP Business One Web Services into the project. The IDE prompts you with a default location of the Web Services. Verify the location of the WS’s and click Finish.
The IDE will iterate over all Web Services to generate the references as part of your project. The progress bar is displayed:

![Add SAP Web Service Clients to LizardApp dialog box with progress bar and list of added services.](image)
Given the large number of Web Services that the IDE is processing, it may take several minutes to generate all the references. You can also run the process in the background. After completion, you should see all the WS References listed under project's Web Service References area:

Once the Web Service processing is complete, you can use the generated WS References to integrate your visual portion of the application. For that, double click on any of the visual components such as button and simply drag and drop the operations into the widget's action method. Make sure to start with Login as it returns a Session ID that is necessary to pass to the rest of the methods.

Note: The current release of the Business One Web Services, October 2007, have been tested with SOAP 1.2, therefore when building Web application make sure to select Web Service Reference corresponding to SOAP 1.2. Additionally check that the service responds with the correct SOAP message version.

You can import other Web Services, such as Google, Yahoo, etc, using the out of the box IDE support for Web Services. For that you can simply select the project, right-mouse-click, then select New->Create Web Service Client and specify URL of the service.
Now the Business One Web Service Clients can be tied to the visual components added to the application earlier. Double click on a widget to process its action, the IDE automatically brings you to the action method of that component. You will see that the Java View of the source code is now displayed. Drag and drop the corresponding Web Service operation into the component action method, use the SOAP 1.2 operations.

Upon drag and drop operation, then double click on the widget to process its action. The IDE auto-generates the corresponding WS client code. This includes

1. Definition of the a service instance

```java
@WebServiceRef(wsdlLocation = "WEB-INF/wsdl/client/LoginService/LoginService.wsdl")
private LoginService service;
```

Since the auto-generated code assigns a non-explicit variable name, it’s recommended to change the variable name to a more meaningful application specific one, i.e.

```java
private LoginService loginService;
```
2. Auto-generation of the function code needs to be filled in with actual values:

You can now populate parameters with the values and execute the application. The IDE default runtime environment is Glassfish, which is an open source Java EE application server. You can preview application output by selecting Output at the bottom of the IDE window and navigating to the Glassfish tab.

See Appendix section for details of the sample Web Services usage.

**Working with Business One WSDLs**

During WSDL processing NetBeans maps complex types to the Java types by default. In a way this is quite helpful as all attributes are auto-generated and enlisted in your code. In case of BusinessOne, some of the modules take long list of input parameters, which ideally should be mapped to a wrapper class, i.e. complex type, that encapsulates these attributes. Thus, if you prefer to work with complex types, you can modify the WSDL properties within the IDE by disabling the default Wrapper style. In this case a complex type defined in the WSDL is mapped to a complex Java class. In order to disable the Wrapper style, right mouse click on a Web Service Client and select Edit Web Service Attributes. Select WSDL customization tab and under Global Customization select the Enable Wrapper Style check box so that the checkbox appears unchecked.
Save the settings. The Information dialog box pops up asking to refresh the WS Client. Select OK.

After disabling Wrapper Style, drag and drop the LoginService's login operation to the application. You'll see that Login and LoginResponse wrapper classes are created instead of a long list of attributes appearing in your method. Individual parameters such as DatabaseType and DatabaseServer are enlisted in the editor popup menu.

**Note:** The LoginService may have issues with Wrapper Style enabled. If that the case, disable the Wrapper Style.

With complex Java type, the code becomes somewhat more compact and easy to navigate.
**WSDL Enumeration Types**

SAP Business One WSDLs define Enum Types that are not compliant with Java Architecture for XML Binding (JAXB) 2.0. Thus, during the WSDL processing, the xml-to-java binding does not map enumeration types to the corresponding Java Enum types. To assist with Business One pre-set Enum values you can access supported enumeration values in the commented section of the auto-generated source code.

Select a type, say Login, and right mouse click on it to Navigate -> Go To Source.
You can preview enumeration values as originally defined in the WSDL. You can either selectively define values for your web application, or create classes that define those values as Java Enum types and are accessible for the entire application. This can be achieved with the ApplicationBean discussed below.
**Passing Session ID Within the User Session**

The default NetBeans Web application is created with three types of Beans – RequestBean, SessionBean, ApplicationBean, which are used to store data or perform necessary processing within a scope of a request, session, or during the application lifetime. In our case, the user session will require passing SessionID that was initially retrieved by the Login() operation can be easily achieved with SessionBean. In the Login method, simply invoke saveData() method on the SessionBean:

```java
this.getSessionBean1().saveData("SessionID", result.getSessionID());
```

In the consequent operations retrieve the session ID with the retrieveData() operation:

```java
String sid = (String)this.getSessionBean1().retrieveData("SessionID");
```

After the session is complete, simply invoke Logout Web Service operation.

You can learn more about the IDE features at the [NetBeans site](http://netbeans.org).
Tracing DI Server SOAP Messages with Java Message Handler

During development process it is helpful to intercept incoming and outgoing SOAP messages sent between the application and the DI Server. This can be easily achieved with the Web Service Message Handler. To create the handler, right mouse click on the project, select New -> Other, which is the very last entry in the list. In the New File window select Web Services from the left panel and Message Handler from the list on the right panel.
Select Next, modify name of the class to MessageHandler and select the package name.

Continue with Finish. You can now implement the handler code yourself or copy and paste the code below into your newly created MessageHandler.

**Java Message Handler**

```java
package lizardapp;

import java.util.Collections;
import java.util.Set;
import javax.xml.namespace.QName;
import javax.xml.soap.SOAPMessage;
import javax.xml.ws.handler.MessageContext;
import javax.xml.ws.handler.soap.SOAPHandler;
import javax.xml.ws.handler.soap.SOAPMessageContext;
import javax.xml.soap.SOAPException;
import java.io.ByteArrayOutputStream;

/**<p>
 * This class enables tracing of the incoming and outgoing SOAP messages
 */

public class MessageHandler implements SOAPHandler<SOAPMessageContext> {
```
public boolean handleMessage(SOAPMessageContext messageContext) {

    System.out.println("MessageHandler.handleMessage");
    boolean outbound = (Boolean)messageContext.get(MessageContext.MESSAGE_OUTBOUND_PROPERTY);
    if (outbound) {
        System.out.println("Direction=outbound");
    } else {
        System.out.println("Direction=inbound");
    }

    SOAPMessage msg = messageContext.getMessage();
    dumpSOAPMessage(msg);

    return true;
}

public Set<QName> getHeaders() {
    return Collections.EMPTY_SET;
}

// Handling Fault
public boolean handleFault(SOAPMessageContext messageContext) {
    System.out.println("ClientSOAPHandler.handleFault");
    boolean outbound = (Boolean)messageContext.get(MessageContext.MESSAGE_OUTBOUND_PROPERTY);
    if (outbound) {
        System.out.println("Direction=outbound");
    } else {
        System.out.println("Direction=inbound");
    }
    if (!outbound) {
        try {
            SOAPMessage msg = ((SOAPMessageContext) messageContext).getMessage();
            dumpSOAPMessage(msg);
            if (messageContext.getMessage().getSOAPBody().getFault() != null) {
                String detailName = null;
                try {
                    detailName = messageContext.getMessage().getSOAPBody().getFault().getDetail().getFirstChild().getLocalName();
                    System.out.println("Fault: detailName="+detailName);
                } catch(Exception e) {
                    e.printStackTrace();
                }
            }
        } catch (SOAPException e) {
            e.printStackTrace();
        }
    }
    return true;
}

public void close(MessageContext context) {
}
private void dumpSOAPMessage(SOAPMessage msg) {
    if (msg == null) {
        System.out.println("SOAP Message is null");
        return;
    }
    System.out.println("\n");
    System.out.println("--------------------");
    System.out.println("DUMP OF SOAP MESSAGE");
    System.out.println("--------------------");
    try {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        msg.writeTo(baos);
        System.out.println(baos.toString(getMessageEncoding(msg)));
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private String getMessageEncoding(SOAPMessage msg) throws SOAPException {
    String encoding = "utf-8";
    if (msg.getProperty(SOAPMessage.CHARACTER_SET_ENCODING) != null) {
        encoding = msg.getProperty(SOAPMessage.CHARACTER_SET_ENCODING).toString();
    }
    return encoding;
}
This module intercepts SOAP messages received and sent from/to the DI Server.

To attach the MessageHandler to the Web Service Client, right mouse click on a WS Client, e.g. LoginService, and select Configure Handlers entry.
Click on Add button and navigate to the MessageHandler class.

Click OK on both windows. You may need to restart the application server.
You can now monitor incoming and outgoing SOAP messages. In the existing project, try to invoke the LoginService WS again after the MessageHandler has been added. The NetBeans IDE tab of the Output area at the bottom of the IDE window contains GlassFish tab. GlassFish application server is a free and open source Java EE engine, that we use to deploy the web application, see glassfish.dev.java.net for more details. You can see the output of the incoming and outgoing SOAP message within that tab.
Below are the actual SOAP messages that are captured by the MessageHandler:

Direction=outbound

DUMP OF SOAP MESSAGE

```
```

Direction=inbound

DUMP OF SOAP MESSAGE

```
```

As you can see, the Message Handler captures both incoming SOAP request and outgoing SOAP response.

**SOAPFault Processing**

Should an application throw an exception, the DI Server returns SOAPFault, which is then translated to the corresponding Java exception. In some cases, DI Server returns the SOAPFault that does not contain a fully qualified sub-code as part of the SOAP 1.2 message returned. The NetBeans IDE enforces SOAP 1.2 specification which requires sub-code as part of the SOAPFault message. In order to receive the application level exception in both Java and .NET applications, you need to modify the B1WSHandler.cs file shipped with Business One Web Services bundle. This module intercepts the SOAP messages received from the DI Server and in some cases edit incoming and outgoing messages. The B1WSHandler.cs file is located under the following directory:

```
C:\Program Files\SAP\SAP Business One Web Services\B1WS\App_Code
```

The `if` clause below re-formats the SOAPFault to become SOAP 1.2 compliant.

```
// Appending prefix for the SOAP 1.2 Subcode to be namespace-qualified

if (res.Contains("Fault")){
```
res = res.Replace("<env:Subcode><env:Value>",
"<env:Subcode><env:Value>env:"");
}

In addition to the Fault handling, B1WSHandler should change the SOAP Envelope header declaration to comply with SOAP 1.2 specification, expected on the Java side. The envelope has to have a namespace of "http://schemas.xmlsoap.org/soap/envelope/". For that the following \textit{if} clause is included:

\begin{verbatim}
if (Request.ContentType.StartsWith("text/xml") &&
    res.Contains("http://www.w3.org/2003/05/soap-envelope"))
{
    res = res.Replace("http://www.w3.org/2003/05/soap-envelope",
                      "http://schemas.xmlsoap.org/soap/envelope/");
}
\end{verbatim}

Both \textit{if} clauses should be added within the \textit{try-catch} block of the B1WSHandler file. You should have the following snippet after updating the handler source code:

\begin{verbatim}
//Call DI Server
try
{
    res = DIServerNode.Interact(request);

    // Replacing SOAP Envelope header to comply with SOAP 1.2 spec
    if (Request.ContentType.StartsWith("text/xml") &&
        res.Contains("http://www.w3.org/2003/05/soap-envelope"))
    {
        res = res.Replace("http://www.w3.org/2003/05/soap-envelope",
                         "http://schemas.xmlsoap.org/soap/envelope/");
    }

    // Appending prefix for the the SOAP 1.2 Subcode to be namespace-qualified
    if (res.Contains("Fault"))
    {
        res = res.Replace("<env:Subcode><env:Value>",
                          "<env:Subcode><env:Value>env:"");
    }
}

catch (Exception ex)
{
    res = ex.Message;
}
\end{verbatim}
Now the SOAPFault will contain a fully qualified Sub-code as part of the SOAP message and will be processed adequately on the Java side. The exception stack will contain the corresponding error message once you printStackTrace of that exception.

```java
} catch (Exception ex) {
    ex.printStackTrace();
}
```

For example, an error of missing some input parameters may result in the following:

```
javax.xml.ws.soap.SOAPFaultException: Accounts receivable/payable is missing [OCRD.DebPayAcct]
```

Here is a complete listing for the .NET B1WSHandler.cs file:

```csharp
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;
using System.Web.Hosting;

/// <summary>
/// HttpHandler in charge of redirecting all B1WS calls to B1 DI Server
/// </summary>
public class B1WSHandler : IHttpHandler
{
    /// <summary>
    /// DI Server Node
    /// </summary>
    private SBODI_Server.Node DI_ServerNode;

    /// <summary>
    /// Create an instance of the DI Server Node
    /// </summary>
    public B1WSHandler()
    {
        DI_ServerNode = (SBODI_Server.Node)new SBODI_Server.Node();
    }
```
/// <summary>
/// Method processing all B1WS requests and redirecting them to DI Server
/// </summary>
/// <param name="context">HttpContext containing all information about the request and response</param>
public void ProcessRequest(HttpContext context)
{
    //read request
    HttpRequest Request = context.Request;
    HttpResponse Response = context.Response;
    System.IO.Stream str = Request.InputStream;
    StreamReader reader = new StreamReader(str);
    string request = reader.ReadToEnd();
    string res = string.Empty;

    // Remove encoding="utf-8" => .NET adds it and DI Server doesn't accept it!
    int index = request.IndexOf("encoding="utf-8");
    if (index > 0 && index < 50)
        request = request.Remove(index, 16);

    //call DI Server
    try
    {
        res = DIServerNode.Interact(request);

        if (Request.ContentType.StartsWith("text/xml") &&
        res.Contains("http://www.w3.org/2003/05/soap-envelope")
        {
            res = res.Replace("http://www.w3.org/2003/05/soap-envelope",
            "http://schemas.xmlsoap.org/soap/envelope/\n",
        }
    }
    catch (Exception ex)
    {
        res = ex.Message;
    }

    // return the response to the caller
    // same content type as the request
    Response.Clear();
    Response.AppendHeader("Content-Type", Request.ContentType);
    Response.Write(res);
public bool IsReusable
{
    // To enable pooling, return true here.
    // This keeps the handler in memory.
    get { return false; }
}

It's worth noting though that intercepting and processing of the incoming or outgoing SOAP messages cause somewhat of performance overhead.
**Conclusion**

In summary, this document outlines a straightforward and fairly easy way to create Web applications using NetBeans IDE and SAP BusinessOne Web Services. There are three main tasks behind creating the Web applications. The first step involves creating a visual web application using the NetBeans GUI editor. The second one is importing the WSDLs into the IDE. And finally, the third step is to drag and drop WS operations into the widget action method. After setting corresponding values into the WS operations, you can deploy the application under Glassfish or Tomcat runtime engines. For troubleshooting purposes, the incoming and outgoing SOAP messages can be accessed with Java Message Handler class. Extending the .NET B1WSHandler module can enable corresponding SOAP Fault messages.
Appendix A

Login Service

JSF Design:
1. Add a button "Login"

In the “Login_action”:
1. Drag and drop the LoginServiceSoap12/Login

1. Definition
@WebServletRef(wsdlLocation = "WEB-INF/wsdl/client/LoginService/LoginService.wsdl")
private LoginService loginService;
2. Client code to login
public String btnLogin_action() {
    try {
        // Call Web Service Operation
        b1wtest.ws.loginservice.LoginServiceSoap port =
        loginService.getLoginServiceSoap12();
        // TODO initialize WS operation arguments here
        java.lang.String databaseServer = "(local)";
        java.lang.String databaseName = "SBODemoUS";
        java.lang.String databaseType = "dst_MSSQL2005";
        java.lang.String databaseUserName = "sa";
        java.lang.String databasePassword = "Initial0";
        java.lang.String companyUsername = "manager";
        java.lang.String companyPassword = "manager";
        java.lang.String language = "ln_English";
        java.lang.String licenseServer = "";
        // TODO process result here
        java.lang.String result = port.login(databaseServer, databaseName,
        databaseType, databaseUserName, databasePassword, companyUsername, companyPassword,
        language, licenseServer);
        this.getSessionBean1().saveData("SessionID", result);
        this.setSessionID(result);
        System.out.println("Result = " + result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
        System.out.println(ex.getMessage());
    }
    return null;
}
Appendix B

Add a business partner with addresses via BusinessPartnersService

JSF Design:
1. Add the labels and textfield for CardCode, CardName, Address 1 and Address 2
2. Add a button “Add BP”

In the “addBP_action”:

Sample code:
```java
public String btnAddBP_action() {
    try {
        // Call Web Service Operation
        bpService.getBusinessPartnersServiceSoapSoap12();
        // TODO initialize WS operation arguments here
        b1ws.test.ws.businesspartnersservice.Add parameters = new
        b1ws.test.ws.businesspartnersservice.Add();
        //b1ws.test.ws.businesspartnersservice.BusinessPartner bp = new
        b1ws.test.ws.businesspartnersservice.BusinessPartner();
        obp = new b1ws.test.ws.businesspartnersservice.BusinessPartner();
        obp.setCardCode(this.getCardCode());
        obp.setCardName(this.getCardName());

        //add 2 addresses: address1 and address2
        BPAddresses addresses = new BPAddresses();
        BAddress addr1 = new BAddress();
        addr1.setAddressName(this.getAddress1());
        addresses.getBAddress().add(addr1);
        BAddress addr2 = new BAddress();
        addr2.setAddressName(this.getAddress2());
        addresses.getBAddress().add(addr2);
        obp.setBPAddresses(addresses);
        parameters.setBusinessPartner(obp);
        b1ws.test.ws.businesspartnersservice.MsgHeader requestHeader = new
        b1ws.test.ws.businesspartnersservice.MsgHeader();
        String sid = (String) this.getSessionBean1().retrieveData("SessionID");
        requestHeader.setSessionID(sid);
        requestHeader.setServiceName("BusinessPartnersService");
        // TODO process result here
        b1ws.test.ws.businesspartnersservice.AddResponse result =
        port.add(parameters, requestHeader);
        System.out.println("Result = " + result);
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
    return null;
}
```
Appendix C

Update the BP with BusinessPartnersService

1. Add a button for “Update BP”.
3. Drag & Drop BusinessPartnersServiceSoap12.Add()

Sample Code:

```java
public String btnUpdateBP_action() {
    // TODO: Process the action. Return value is a navigation case name where null will return to the same page.
    try { // Call Web Service Operation
        b1wstest.ws.businesspartnersservice.BusinessPartnersServiceSoap port = bpService.getBusinessPartnersServiceSoap12();
        // TODO initialize WS operation arguments here
        b1wstest.ws.businesspartnersservice.GetByParams parameters = new b1wstest.ws.businesspartnersservice.GetByParams();
        b1wstest.ws.businesspartnersservice.BusinessPartnerParams bpParms = new b1wstest.ws.businesspartnersservice.BusinessPartnerParams();
        bpParms.setCardCode(this.getCardCode());
        parameters.setBusinessPartnerParams(bpParms);
        b1wstest.ws.businesspartnersservice.MsgHeader requestHeader = new b1wstest.ws.businesspartnersservice.MsgHeader();
        requestHeader.setSessionID(this.getSessionID());
        requestHeader.setServiceName(bpService.getServiceName().getLocalPart());
        // TODO process result here
        b1wstest.ws.businesspartnersservice.GetByParamsResponse result = port.getByParams(parameters, requestHeader);
        obp = result.getBusinessPartner();
        System.out.println("Result = "+ result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
        System.out.println(ex.getMessage());
    }
    try { // Call Web Service Operation
        b1wstest.ws.businesspartnersservice.BusinessPartnersServiceSoap port = bpService.getBusinessPartnersServiceSoap12();
        // TODO initialize WS operation arguments here
        b1wstest.ws.businesspartnersservice.Update parameters = new b1wstest.ws.businesspartnersservice.Update();
        obp.setCardCode(this.getCardCode());
        obp.setCardName(this.getCardName());
        //add 2 addresses: address1 and address2
        BPAddresses addresses = new BPAddresses();
        BPAddress addr1 = new BPAddress();
        addr1.setAddressName(this.getAddress1());
        addresses.getBPAddress().add(addr1);
        BPAddress addr2 = new BPAddress();
        addr2.setAddressName(this.getAddress2());
        addresses.getBPAddress().add(addr2);
        obp.setBPAddresses(addresses);
    }
}
```
parameters.setBusinessPartner(obp);
b1wstest.ws.businesspartnersservice.MsgHeader requestHeader = new
b1wstest.ws.businesspartnersservice.MsgHeader();
requestHeader.setSessionID(this.getSessionID());
requestHeader.setServiceName(bpService.getServiceName().getLocalPart());
// TODO process result here
b1wstest.ws.businesspartnersservice.UpdateResponse result =
port.update(parameters, requestHeader);
    System.out.println("Result = " + result);
} catch (Exception ex) {
    // TODO handle custom exceptions here
    System.out.println(ex.getMessage());
}

return null;
Appendix D

Add a A/R Invoice via InvoiceService

1. Add a button for "Add Inv",
2. Drag & Drop InvoiceServiceSoap12. Add()

Sample Code:
```java
public String btnAddInv_action() { 
    try { // Call Web Service Operation
        b1wstest.ws.invoicesservice.InvoicesServiceSoap port = 
            invoiceService.getInvoicesServiceSoap12();
        // TODO initialize WS operation arguments here
        b1wstest.ws.invoicesservice.Add parameters = new
        b1wstest.ws.invoicesservice.Add();
        b1wstest.ws.invoicesservice.Document oInvoice = new
        b1wstest.ws.invoicesservice.Document();
        oInvoice.setCardCode(this.getCardCode());
        XMLGregorianCalendar docDueDate = new XMLGregorianCalendarImpl();
        docDueDate.setYear(2008);
        docDueDate.setMonth(10);
        docDueDate.setDay(16);
        oInvoice.setDocDueDate(docDueDate);
        //oInvoice.setDocType("dDocument_Items");
        b1wstest.ws.invoicesservice.Document.DocumentLines oLines = new
        b1wstest.ws.invoicesservice.Document.DocumentLines();
        b1wstest.ws.invoicesservice.DocumentLines.DocumentLine line1 = new
        line1.setItemCode("A00001");
        line1.setUnitPrice(567.00);
        line1.setQuantity(2.00);
        line1.setTaxCode("PA");
        oLines.getDocumentLine().add(line1);
        oInvoice.setDocumentLines(oLines);
        parameters.setDocument(oInvoice);
        b1wstest.ws.invoicesservice.MsgHeader requestHeader = new
        b1wstest.ws.invoicesservice.MsgHeader();
        requestHeader.setSessionID(this.getSessionID());
        b1wstest.ws.invoicesservice.AddResponse result = port.add(parameters, 
            requestHeader);
        System.out.println("Result = " + result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
        System.out.println(ex.getMessage());
    }

    return null;
}
```
Related Content

- NetBeans Features
- NetBeans Wiki
- Web Services with NetBeans
- Glassfish Application Server
- SAP Business One 2007
- SAP Developer Network Business One Resources
- SAP Business One Web Services (B1WS) Tool

For more information, visit the Java homepage.
Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.