REPORT: 3RD ANNUAL WEB SERVICES EDGE CONFERENCE & EXPO PAGE 42

WebSphere® DEVELOPER'S JOURNAL

The World's Leading Independent WebSphere Developer Resource

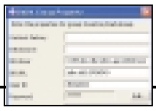
WebSphereDevelopersJournal.com

APRIL VOLUME 2 ISSUE 4

DISPLAY UNTIL JUNE 30, 2003
$8.99US $9.99CAN
04>
0 09281 03422 3

SYS-CON MEDIA

The WAS 5 Web Services Technology Preview
Get a head start on the J2EE 1.4 APIs



VERSATA

WWW.VERSATA.COM/BUSINESSLOGICDESIGNER

WILY TECHNOLOGY

WWW.WILYTECH.COM

Perspectives on the World of Technology

BY JACK MARTIN

Over the course of the past couple of weeks I have had a most unusual experience. I have spent a good portion of my time working with the people at Candle Corporation. I typically divide my time between looking at the world of technology from my own perspective – and through the lens of IBM as it deals with the multitude of people and products it is composed of. Until this month I had never given much thought to how an independent software vendor differs from IBM. I was surprised to find out that Candle has been in business for over 27 years and has spent a substantial portion of that time as a research and development partner with IBM.

What I was exposed to was a lean group of dedicated people with the single-minded mission to get their company's products in front of more CIOs and developers – nothing more, nothing less. I also learned all about real-time monitoring tools. I found that their tools actually work as described; they ship fully functional products to their customers – without endless adjustments or excuses. The people in Redmond could learn a thing or to from this group.

On another note, I heard from some readers in response to my editorial last month. They are convinced there are no land and home bargains in the United States because they can find no mention of it on the Internet. It appears that they have confined their worldview to whatever their browser or e-mail client can show them; quite a virtual and misguided view. One must remember that drinking the Kool-Aid of the technology industry, which says that all information will come to us via a networked world – wired or wireless – is wrong, and also completely impossible.

Stop and think about it for a moment. If you lived in a place that almost everyone you have ever known has left behind, and the land is being reabsorbed by nature, would the first thing you did be to set up a Web site to announce the event and the available real estate opportunities to any interested Web surfer? Or possibly you'd set up a news group like alt.dying-towns and maybe develop a spam program to make sure everyone that uses the Internet is kept fully abreast of what is going on in your invisible part of the world. As far as we know, the entire universe, with the exception of our planet, operates without the Internet. Stars are being born every day as others die, with not a mention of the Internet, or for that matter, anything else on the planet. Perhaps it's not happening at all, as there is no Webcam showing us the latest events.

Which brings me to my final thought this month. The entire point of this next wave of technological innovation is about finally beginning to deliver on the original intent of the technological revolution: modeling and reflecting human behavior in ways that better serve society and individuals' wants, needs, and desires. As we enter this new world we are finding that machines are becoming more and more the invisible hand that we have always secretly desired them to be, helpful servants waiting to fulfill our needs in both our business and personal lives, needing nothing but electricity to keep mindlessly humming away. As this progressive wave of automation continues to its ultimate goal of seamless task accomplishment, our society's collective memory of how things are done now will begin to fade – just as did the memory of how automobiles once had to be cranked to life by hand. One thing that will never change is the joining together of people to accomplish a common goal the way the people at Candle do every day.

ABOUT THE AUTHOR... Jack Martin, editor-in-chief of ***WebSphere Developer's Journal***, is cofounder and CEO of Simplex Knowledge Company, an Internet software boutique specializing in WebSphere development. Simplex developed the first remote video transmission system designed specifically for childcare centers, which received worldwide media attention, and the world's first diagnostic-quality ultrasound broadcast system. E-MAIL... jack@sys-con.com

WebSphere® DEVELOPER'S JOURNAL

ADVISORY BOARD
Richard Arone, David Caddis, Michael Clevestig, Devi Gupta, Lloyd Hagemo, Barbara Johnson, James Martin, Doug Stephen, Victor Valle

Editor-in-Chief	JACK MARTIN
Editorial Director	JEREMY GEELAN
Executive Editor	GAIL SCHULTZ
Contributing Editor	PATTI MARTIN
Product Review Editor	JAY JOHNSON
Special Projects Editor	RICHARD ARONE
Managing Editor	JEAN CASSIDY
Editor	NANCY VALENTINE
Associate Editors	JAMIE MATUSOW JENNIFER STILLEY

WRITERS IN THIS ISSUE
James Carey, Brent Carlson, Denise Gabardo, Joan Haggarty, Bassem Jamaleddine, Adam Kolawa, Christina Lau, Jack Martin, Simon Pepper, Deependra Shrestha, Raghu Shrestha, Soutik Singha, André Tost

SUBSCRIPTIONS
For subscriptions and requests for bulk orders, please send your letters to Subscription Department.
SUBSCRIBE@SYS-CON.COM
Cover Price: $8.99/Issue Domestic: $149/YR (12 Issues)
Canada/Mexico: $169/YR Overseas: $179/YR
(U.S. Banks or Money Orders)
Back issues: $15 U.S. $20 All others

PRESIDENT AND CEO	FUAT A. KIRCAALI
SENIOR VP, SALES & MARKETING	CARMEN GONZALEZ
VP, INFORMATION SYSTEMS	ROBERT DIAMOND
VP, SALES & MARKETING	MILES SILVERMAN
PRESIDENT, SYS-CON EVENTS	GRISHA DAVIDA
PRODUCTION CONSULTANT	JIM MORGAN
FINANCIAL ANALYST	JOAN LAROSE
ACCOUNTS RECEIVABLE	KERRI VON ACHEN
ACCOUNTS PAYABLE	BETTY WHITE
ADVERTISING ACCOUNT MANAGERS	MEGAN MUSSA ROBYN FORMA
ASSOCIATE SALES MANAGERS	CARRIE GEBERT ALISA CATALANO KRISTIN KUHNLE
CONFERENCE MANAGER	MICHAEL LYNCH
ART DIRECTOR	ALEX BOTERO
ASSOCIATE ART DIRECTORS	LOUIS F. CUFFARI RICHARD SILVERBERG TAMI BEATTY
WEB DESIGNERS	STEPHEN KILMURRAY CHRISTOPHER CROCE
CONTENT EDITOR	LIN GOETZ
SENIOR CUSTOMER CARE/ CIRCULATION SPECIALIST	NIKI PANAGOPOULOS
CUSTOMER SERVICE REPRESENTATIVE	SHELIA DICKERSON
CUSTOMER RELATIONS MANAGER	RACHEL MCGOURAN

EDITORIAL OFFICES
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
WebSphere® Developer's Journal (ISSN# 1535-6914) is published monthly (12 times a year).
Postmaster send address changes to:
WebSphere Developer's Journal, SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

CANDLE

WWW.CANDLE.COM/WWW1/HENRY1_WSDJ

MQSOFTWARE

WWW.MQSOFTWARE.COM

MQSOFTWARE

WWW.MQSOFTWARE.COM

Standards pave the way

Web Services–Based Application Server Interoperability

BY SIMON PEPPER

The future of application servers, including WebSphere, lies in refining functionality that allows applications to be treated as services. This includes allowing future applications to be reusable as well as interoperable with older standards-based applications, such as CORBA. It is easy to discuss, but the actual act of integrating services can be extremely complicated.

ABOUT THE AUTHOR
Simon Pepper, director of product marketing at IONA (www.iona.com), focuses on IONA's Orbix Application Server Platform, which combines IONA's market-leading CORBA, advanced J2EE, and awarding-winning Web services technologies. Simon has over 10 years of experience in the field of distributed computing. Simon is IONA's representative on the Java Community Process Executive Committee.

E-MAIL
simon.pepper@iona.com

This article will discuss interoperability between Web services, J2EE, and CORBA from Windows, through Unix to the mainframe, highlighting:

- The exposure of Enterprise JavaBeans (EJBs) as Web services
- The exposure of CORBA objects as Web services
- The interoperability of CORBA and Enterprise JavaBeans: CORBA calling EJBs, EJBs calling CORBA
- Transactional interoperability, which becomes increasingly important as more computer languages are developed
- Security (including HTTPS, SAML/XKML, CSIv2, TLS/SSL, RACF), which also becomes an issue as the technologies are used to transfer sensitive materials that need to remain private

Three important technologies exist that are able to support the concept of distributed applications based on broadly accepted standards. Those are Web services and .NET, Enterprise JavaBeans/J2EE, and CORBA. CORBA and EJBs are already used as the core of enterprise-level application systems. Web services and .NET are beginning to be used for B2B-applications across enterprise boundaries and to connect applications across different departments. Obviously none of those technologies will completely dominate the software industry, thus interoperability becomes one of the most crucial issues. The focus, therefore, is on how interoperability is achieved among those technologies.

While Enterprise JavaBeans and CORBA are already used in many companies as the basis for distributed applications, Web services is gaining traction in those environments. Currently a number of large companies are evaluating whether this technology is a viable solution for the integration problem inside and across the firewall. For these companies, Web services could introduce a level of abstraction, so that diverse applications are made accessible in a uniform way.

Since you usually find a heterogeneous system environment in large enterprises, it seems useful to have a look at the different integration scenarios in detail:

- Possible clients for a Web service
- An EJB calls a CORBA object
- A CORBA client uses an EJB
- A CORBA service is provided as a Web service
- An EJB is published as a Web service

Possible Clients for a Web Service

In order to keep the promise of solving the integration problem for distributed systems in a simple and standards-based way, Web services need to be accessible by a great variety of "clients" (as a role within an interaction). To illustrate this, Listing 1 shows how a C# client invokes a credit card validation service implemented as a Web service. The FNBCreditCardValidationService.cs file contains the code shown in Listing 2.

Another important class of clients is Java based. An integration platform generates both a proxy client for the Java 2 Standard Edition (J2SE) as well as a static client for the Java 2 Micro Edition (J2ME), based on the description of an interface as WSDL (Web Services Definition Language), which can be directly used or serve as a template. Listing 3 shows what a Java client for a Web service looks like using J2SE.

The Method doSoapIO executes the SOAP request as shown in Listing 4.

An EJB Calls a CORBA Object

In order to invoke a CORBA server out of an EJB given a J2EE implementation with the appropriate connectivity, all that is needed is to generate the CORBA "stubs" using the IDL Compiler (which is a part of the CORBA implementation). If the J2EE application server is based on a CORBA infrastructure, it is possible to use the same ORB (Object Request Broker) that manages the EJB. In this case it would be possible to exchange a transaction context transparently between the EJB container and the CORBA server, thus enabling a transactional secured communication.

This means that a distributed transaction between the EJB container and CORBA OTS (Object Transaction Server) can be managed using a two-phase commit protocol. The following example is based on the assumption that the IOR (Interoperable Object Reference) of the CORBA server is stored within a CORBA Naming Service under DirectoryEnquiries and that this Naming Service is known to the EJB container. If the IDL of the CORBA server looks like the code in Listing 5, the corresponding code inside a method of a session bean would look like the code in Listing 6.

A CORBA Client Uses an EJB as a Service

To generate a CORBA interface for an EJB remote interface an rmic (RMI Compiler) could be used, but the result is not very elegant because of the excessive usage of "object by value" in the RMI-IIOP standard. This becomes especially problematic when a procedural programming language is used for the invoking program. Therefore it seems more practical if between the CORBA client and the EJB, a bridge is used that acts as a CORBA server and encapsulates an EJB client. This CORBA-EJB bridge implements a CORBA-IDL, which is designed according to the corresponding remote interface and could eventually even be executed in-process (depending on the application server capabilities).

An EJB/CORBA Service Is Published as a Web Service

To publish an EJB or a CORBA server as a Web service, a tool is needed to generate WSDL based on the remote interface of the EJB or CORBA IDL and that is capable of receiving a SOAP message and acting as a client for the EJB or CORBA server. As the logic of those two cases looks remarkably similar, the example shown in Listing 7 uses an EJB. It is a session bean with two methods: inchToMM (inches to millimeters) and mmToInch (millimeters to inches). The corresponding WSDL looks like the code shown in Listing 8.

An internal SOAP listener would be used during runtime to act as an EJB client. A session bean would play the role of a CORBA client for a CORBA server.

Conclusion

Web services is a promising technology that can be viewed as the technological groundwork for facilitating a service-oriented architecture.

Obviously, enterprises will still use a variety of different technologies, and Web services can bridge the existing boundaries. Integration has a number of aspects, and Web services can play a crucial role when it comes to working with coarse-grained services. As shown in this article, you can use different techniques to integrate CORBA and EJBs, which seems to be reasonable when components need to be integrated in order to participate in a distributed transaction or to achieve maximized performance or flexibility.

Currently, the standards have been defined for transaction processing using Web services, and the business transaction technical committee of OASIS (Organization for the Advancement of Structured Information Standards) is working on the BTP (Business Transaction Protocol) specification. As work on WS-Transactions is under way, in the near future the software industry will agree upon a number of standards based on Web services that will enable the integration of software systems in ways that will bring a new quality to usage of the Internet.

Acknowledgment

The author acknowledges Arne Koschel and Klaus Hofmann zur Linden of IONA Technologies for their contributions to this article.

LISTING 1:
```
int num = int.Parse(shortnum);
int amnt = int.Parse(txt_amount.Text);

// How to use the Web service in 2 easy steps:
// 1. Connect to the validation service
FNBCreditCardValidationService myClass = new
FNBCreditCardValidationService ();
// 2. Call the validateCard method defined on this service
short code =0;
try
{
   code = myClass.validateCard (num, amnt);
}
catch (System.Net.WebException ex)
{
   Console.WriteLine ("Caught Exception" + ex);
   return;
}
   lbl_code.Text = code.ToString();
   cmd_confirm.Enabled = true;
}
```

LISTING 2:
```
using System.Diagnostics;
using System.Xml.Serialization;
using System;
using System.Web.Services.Protocols;
using System.Web.Services;

System.Web.Services.WebServiceBindingAttribute

(Name="FNBCreditCardValidationPortBinding",
Namespace="http://xmlbus.com/FNBCreditCardValidation")]
public class FNBCreditCardValidationService :
System.Web.Services.Protocols.SoapHttpClientProtocol {
    [System.Diagnostics.DebuggerStepThroughAttribute()]
public FNBCreditCardValidationService()
{
   this.Url ="http://klaus:9000/xmlbus/container" +
           "/FNBCreditCardValidation/FNBCreditCardValidatio" +
           "nService/FNBCreditCardValidationPort/";
}

System.Diagnostics.DebuggerStepThroughAttribute();
System.Web.Services.Protocols.SoapRpcMethodAttribute("",
   RequestNamespace = "http://xmlbus.com/"

"FNBCreditCardValidation",
ResponseNamespace = "http://xmlbus.com/" +
"FNBCreditCardValidation");
[return:System.Xml.Serialization.SoapElementAttribute
("return")]

    public bool confirmPurchase(int param0, System.Single
    param1, short param2) {
        object[] results = this.Invoke("confirmPurchase",
new object[] {
                      param0,
                      param1,
                      param2});
         return ((bool)(results[0]));
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    public System.IAsyncResult BeginconfirmPurchase(int
    param0, System.Single param1, short param2,
    System.AsyncCallback callback, object asyncState) {
         return this.BeginInvoke("confirmPurchase", new
         object[] {
                      param0,
                      param1,
                      param2}, callback, asyncState);
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    public bool EndconfirmPurchase(System.IAsyncResult
    asyncResult) {
         object[] results = this.EndInvoke(asyncResult);
         return ((bool)(results[0]));
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.Web.Services.Protocols.SoapRpcMethodAttribute("",
RequestNamespace="http://xmlbus.com/FNBCreditCardValidation",
ResponseNamespace="http://xmlbus.com/FNBCreditCardValidation"
)]

    [return:System.Xml.Serialization.SoapElementAttribute
    ("return")]
    public short validateCard(int param0, System.Single
    param1) {
         object[] results = this.Invoke("validateCard", new
         object[] {
                      param0,
                      param1});
         return ((short)(results[0]));
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    public System.IAsyncResult BeginvalidateCard(int param0,
    System.Single param1, System.AsyncCallback callback,
    object asyncState) {
         return this.BeginInvoke("validateCard", new object[]
{
                      param0,
                      param1}, callback, asyncState);
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    public short EndvalidateCard(System.IAsyncResult
    asyncResult) {
         object[] results = this.EndInvoke(asyncResult);
         return ((short)(results[0]));
    }
}
```

LISTING 3:

```
public float inchToMM(float _param0) throws  RemoteException
{
String soapAction = "";
StringBuffer envelope = new StringBuffer(
  "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
envelope.append("<SOAP-ENV:Envelope ");
envelope.append("\n      xmlns:SOAP-
          ENV=\"http://schemas.xmlsoap.org/soap/envelope/\"");
          envelope.append("\n
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"");
envelope.append("\n
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-
                              instance\"");
envelope.append(">\n");
envelope.append("  <SOAP-ENV:Body>\n");
envelope.append("    <m1:inchToMM xmlns:m1=\"urn:target-con-
verter-
service\" SOAP-ENV:encodingStyle=
                    \"http://schemas.xmlsoap.org/soap/encod
                    ing/\">\n");
envelope.append("      <param0 xsi:type=\"xsd:float\">");
envelope.append(floatToString(_param0));
envelope.append("</param0>\n");
envelope.append("    </m1:inchToMM>\n");
envelope.append("  </SOAP-ENV:Body>\n");
envelope.append("</SOAP-ENV:Envelope>\n");
SOAPResponse_Handler handler =
  new SOAPResponse_Handler("urn:target-converter-service",
                       "inchToMMResponse", "return");
doSoapIO(soapAction,envelope,handler);
if (handler.isFault())
{
  throw new RemoteException(handler.getFaultContent());
}
try
{
    String resultString = handler.getResult();
    if (resultString == null) throw new
  RemoteException("Could not find return in soap response");
  return parseFloat(resultString);
} catch (Exception e)
{
  throw new RemoteException
         ("Error generating result." + e.toString());
}
```

LISTING 4:

```
private void doSoapIO(String soapAction, StringBuffer enve-
lope,
                    XMLFilterImpl handler) throws
                    RemoteException
{
. . .
URL url = new URL(soapEndPointURL);
HttpURLConnection connect =
(HttpURLConnection)url.openConnection();
connect.setDoOutput(true);
byte bytes[] = envelope.toString().getBytes("UTF-8");

connect.setRequestProperty("SOAPAction","\""+soapAction+"\"")
;
connect.setRequestProperty("content-type","text/xml");
connect.setRequestProperty("content-length",""+bytes.length);

OutputStream out = connect.getOutputStream();
out.write(bytes);
out.flush();
}
```

LISTING 5:

```
// IDL: EJB_to_CORBA.IDL
module corbaserver {
  struct addressStruct {
         string name;
         string number;
         string street;
         string town;
         string state;
         string zip;
  };
  struct nameStruct {
         string name;
         string number;
  };

  interface ejb_to_corba_int
  {
         void getNumber     (in nameStruct Name,
                              out addressStruct
                              Number);
  };
};
```

LISTING 6:

```
// Declaration of CORBA Variables and init
```

IBM

WWW.IBM.COM/WEBSPHERE/OPENTOOLS

```
org.omg.CORBA.ORB orb = null;
org.omg.CosNaming.NamingContextExt context = null;
javax.naming.Context jndi_context = null;
org.omg.CORBA.Object objRef = null;
Properties myProps;

// Initialization of the CORBA ORB of the EJB
// The ORB names ‘OrbixJ2EE’ is preconfigured with some
// properties like security and transaction propagation
Properties orbProperties = new Properties();
orbProperties.put("org.omg.CORBA.ORBClass",
                    "com.iona.corba.art.artimpl.ORBImpl");
orbProperties.put("org.omg.CORBA.ORBSingletonClass",
com.iona.corba.art.artimpl.ORBSingleton");
String[] orbArgs = new String[] {"-ORBname"," OrbixJ2EE"};
orb = org.omg.CORBA.ORB.init(orbArgs, orbProperties);

corbaserver.ejb_to_corba_int directoryEnquiries = null;
NameComponent[] tmpName = new NameComponent[1];
try
{
  // get reference to Naming Service
  objRef = orb.resolve_initial_references("NameService");
  context = NamingContextExtHelper.narrow(objRef);
  // Create Naming Component
  tmpName[0] = new NameComponent("DirectoryEnquiries", "");
  // get object reference
  objRef = context.resolve(tmpName);
directoryEnquiries=(corbaserver.ejb_to_corba_int)
          corbaserver.ejb_to_corba_intHelper.narrow
                    ((org.omg.CORBA.Object)objRef);
}
catch ( org.omg.CORBA.ORBPackage.InvalidName ex )
    …
}

// complete input structure
nameStruct myName = new nameStruct(in_name, "0");
// create output structure
addressStructHolder ns = new addressStructHolder();
// calling the getNumber method
try
{
  System.out.println("BEAN> Calling getNumber()
                    method on DirectoryEnquiries CORBA serv
                    er object");
  directoryEnquiries.getNumber(myName, ns);
  System.out.println("BEAN> After calling getNumber()");
}
  catch(org.omg.CORBA.SystemException e)
  ...
return ns.value.number;
};
```

LISTING 7:

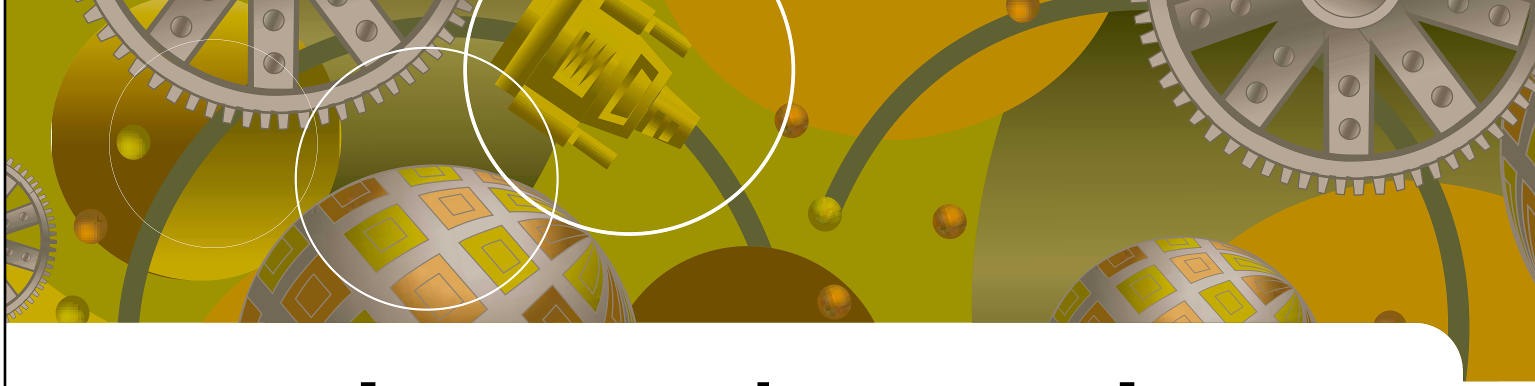
```
public class ConverterBean implements SessionBean {

  public float inchToMM(float inches)
  {
          try {
                    return inches * ((Float)new
                    InitialContext().lookup(

"java:comp/env/inchToMillimeterRate")).floatValue();
          } catch (NamingException ex)
          {
                    throw new EJBException(ex);
          }
  }
  public float mmToInch(float millimeters)
                    throws ConversionException {
          try
          {
                    if (millimeters < 0) throw
                              new
                              ConversionException("Test");
                    return millimeters / ((Float)new
                    InitialContext().lookup(
"java:comp/env/inchToMillimeterRate")).floatValue();
          } catch (NamingException ex) {
                    throw new EJBException(ex);
          }
  }
```

LISTING 8:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="ConverterService"
          targetNamespace="urn:target-converter-service"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="urn:target-converter-service"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd1="http://xmlbus.com/xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <message name="inchToMM">
          <part name="param0" type="xsd:float"/>
  </message>
  <message name="inchToMMResponse">
          <part name="return" type="xsd:float"/>
  </message>
  <message name="mmToInch">
          <part name="param0" type="xsd:float"/>
  </message>
  <message name="mmToInchResponse">
          <part name="return" type="xsd:float"/>
  </message>
  <portType name="ConverterPortType">
          <operation name="inchToMM">
                    <input message="tns:inchToMM"
                    name="inchToMM"/>
                    <output message="tns:inchToMMResponse"

                    name="inchToMMResponse"/>
          </operation>
          <operation name="mmToInch">
                    <input message="tns:mmToInch"
                    name="mmToInch"/>
                    <output message="tns:mmToInchResponse"
                    name="mmToInchResponse"/>
          </operation>
  </portType>

  <binding name="ConverterPortBinding"
  type="tns:ConverterPortType">
  <soap:binding style="rpc"
  transport="http://schemas.xmlsoap.org/soap/http"/>
          <operation name="inchToMM">
                    <soap:operation soapAction=""
                    style="rpc"/>
                    <input name="inchToMM">
                              <soap:body encodingStyle=

"http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:target-converter-service"
                    use="encoded"/>
                    </input>
                    <output name="inchToMMResponse">
                              <soap:body encodingStyle=
                              "http://schemas.xmlsoap.org/
                              soap/encoding/"
                              namespace=
                              "urn:target-converter-service"
                              use="encoded"/>
                    </output>
          </operation>
  </binding>
  <service name="ConverterService">
          <port binding="tns:ConverterPortBinding"
                    name="ConverterPort">
                    <soap:address location="http://local
                    host:8080/xmlbus/container/Converter/
                    ConverterService/ConverterPort"/>
          </port>

  </service>
</definitions>
```

REPORTINGENGINES

WWW.REPORTINGENGINES.COM/DOWNLOAD/WSDJ1.JSP



Creating Web Services from Stored Procedures Using WebSphere Studio

Wizards can take the drudgery out of Web service implementation

– BY JOAN HAGGARTY AND CHRISTINA LAU –

Web services are no longer a new concept. They are rapidly gaining acceptance and use in the development of e-business applications. By now, the benefits of using Web services are clear: they provide a modular, self-describing, and self-contained mechanism to share business logic over the Internet using standardized messaging protocols. Business logic is separated from the client code and the database and can be made available to numerous applications.

ABOUT THE AUTHOR
Joan Haggarty is an advisory software developer at the IBM Toronto Lab. Joan is a lead developer on the WebSphere Studio Application Developer XML tools.

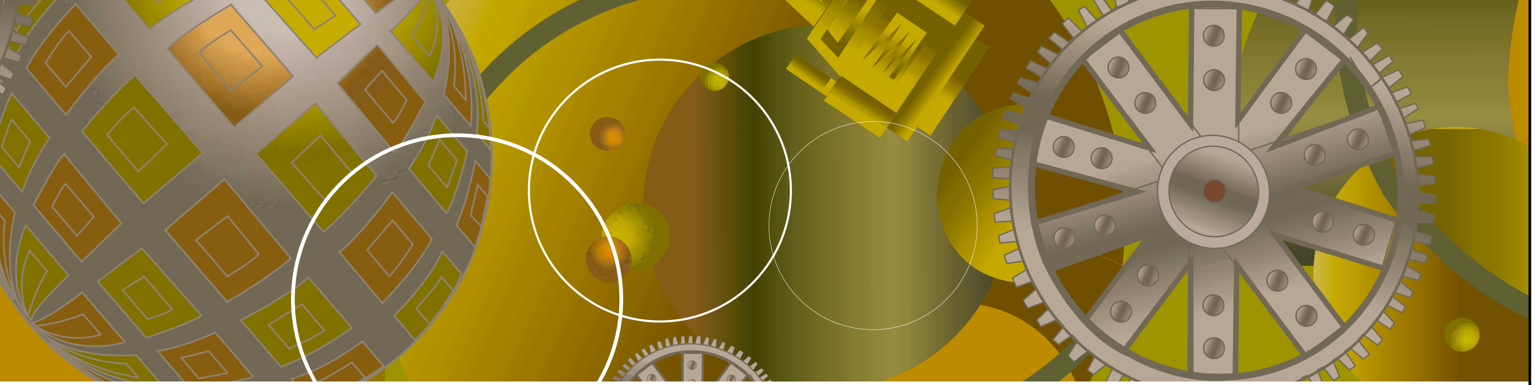
E-MAIL
joan@ca.ibm.com

The payoffs are obvious. However, related standards such as SOAP, AXIS, UDDI, and JAX-RPC are evolving quickly and various components are needed to completely implement a Web service, including client code, message protocols, WSDL, security, registration, and deployment. It is challenging, and in some cases tedious, for developers to assemble the appropriate components from scratch while keeping pace with the changing technology and standards modifications.

IBM WebSphere Studio provides wizards that generate standard Web service code for developers, allowing them to focus on the code specific to their application instead of taking time to write the standard code used for Web service implementation. In this article we will describe how to create an inventory management Web service from a Java stored procedure using WebSphere Studio. We use a stored procedure in this application because the scenario requires multiple SQL statements to be executed based on a computation. By using a stored procedure, we reduce the number of calls made between the application and the database over the network, which can result in substantial performance gains. There are other benefits of using stored procedures in your application. Stored procedures make application updates and maintenance smoother by centralizing business logic. If you make changes to a stored procedure, the changes are immediately available to all client applications that use it. In addition, security and administration are easier when the logic is relegated to the database server since the database management system already takes care of these issues.

The Scenario

We will use an inventory management scenario to demonstrate how to create a Web service from a stored procedure. A satellite store checks to see if there is enough



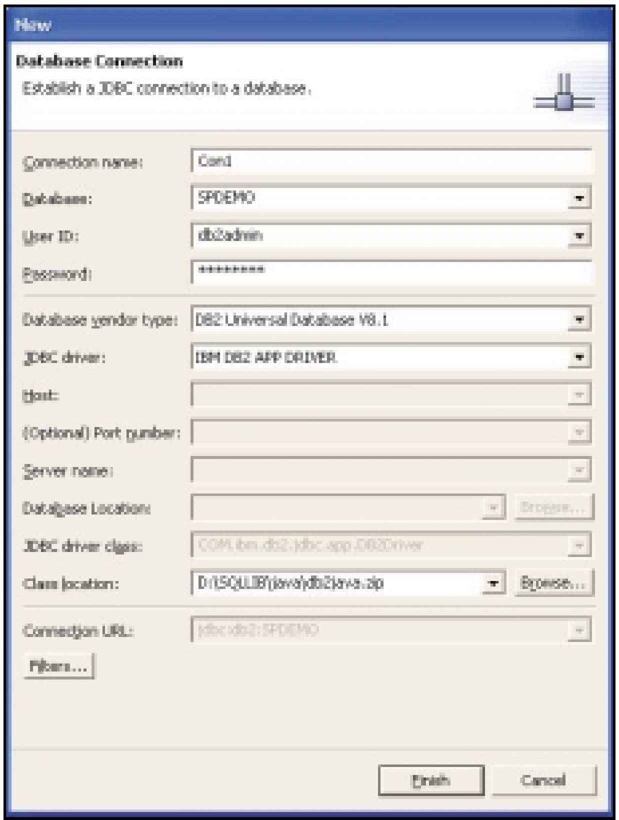
inventory in the warehouse to satisfy a customer order. If there is, inventory records need to be updated to remove the inventory from the warehouse database. A report containing a description of the item and an estimated arrival time is required by the satellite store as confirmation of the product availability.

The stored procedure logic is as follows:

1. Query the inventory table using the item ID to retrieve the current available inventory.
2. Compare the available quantity with the requested quantity.
3. If there is enough quantity to satisfy this request, update the inventory table by subtracting the requested quantity from the available quantity.
4. Query the inventory table to retrieve the product description and the projected delivery date for the item.

The stored procedure will take two parameters:

- ***Quantity:*** The number of items requested by the satellite store
- ***itemId:*** A unique ID that identifies the item



New	
Database Connection	
Establish a JDBC connection to a database.	
Connection name:	Con1
Database:	SPDEMO
User ID:	db2admin
Password:	********
Database vendor type:	DB2 Universal Database V8.1
JDBC driver:	IBM DB2 APP DRIVER
Host:	
(Optional) Port number:	
Server name:	
Database Location:	
JDBC driver class:	COM.ibm.db2.jdbc.app.DB2Driver
Class location:	D:\SQLLIB\java\db2java.zip
Connection URL:	jdbc:db2:SPDEMO

FIG. 1: DATABASE CONNECTION WIZARD

Creating the Stored Procedure

Building and deploying a stored procedure can be a daunting task. WebSphere Studio provides an integrated set of tools to make it much easier. Using the stored procedure tooling in WebSphere Studio, you can:

- Create a new stored procedure
- Modify an existing stored procedure
- Build a stored procedure and register it with the DB2 server
- Run a stored procedure
- Debug a stored procedure
- Drop a stored procedure from the DB2 server

To create a stored procedure, we first need a connection to a database. The Connection wizard is used to establish a connection as shown in Figure 1. It prompts for the relevant database, JDBC information, and optional filters to subset the list of table definitions to import. Once the database artifacts are imported into a project, we launch the Java Stored Procedure wizard to build the Java stored procedure.

Our Java stored procedure is composed of three SQL statements: a SELECT statement to query the inventory available, an UPDATE statement to update the table if there is enough inventory, and another SELECT statement to obtain the product description and an estimated delivery date to return to the client. In the Java Stored Procedure wizard, we will indicate that we want to generate multiple SQL statements as shown in Figure 2. We add the three SQL statements and complete each one using the SQL wizard launched from the SQL Assist button.

Once we are finished with the Java Stored Procedure wizard, a file containing the Java stored procedure will be created. The generated code makes a connection to the database and then executes the three SQL statements. We must customize the generated Java code to add in the logic that performs the availability check and that will update the table to reflect used inventory only if there is enough inventory available to fulfill the request. Listing 1 shows the modified Java stored procedure.

Building and Running the Stored Procedure

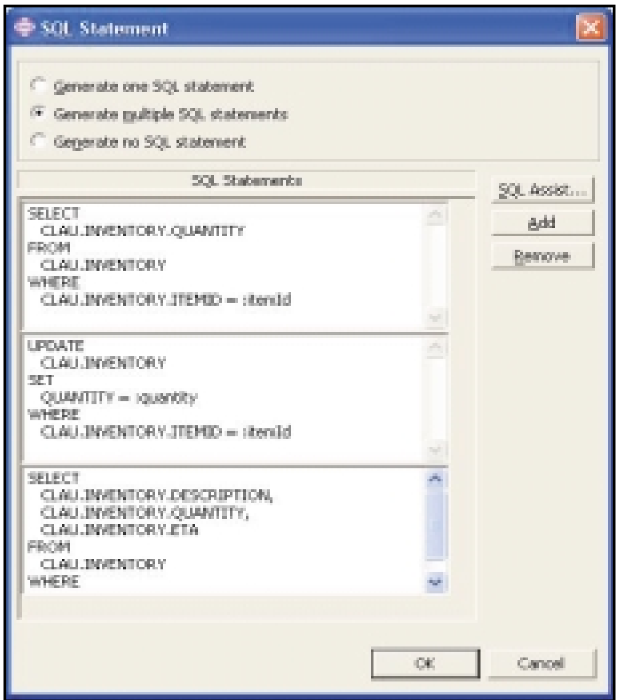
The next step is to build the stored procedure and register it with the DB2 server. To build the stored procedure, select the UPDATEINVENTORY stored procedure in the Data Definition view and choose Build. The Java source code is compiled and the stored procedure is registered with the database server if the compilation is successful. You can view the progress of the build process

ABOUT THE AUTHOR

Christina Lau is a senior technical staff member at IBM. Christina is an architect on the WebSphere Business Scenarios team responsible for improving the integration and consistency across the WebSphere platform using customer scenarios.

E-MAIL
clau@ca.ibm.com

and the messages in the DBOutput view as shown in Figure 3. If there are errors in the Java stored procedure, the lines containing errors will also be indicated in the DB Output view.


SQL Statement
Generate one SQL statement
Generate multiple SQL statements
Generate no SQL statement
SQL Statements
SQL Assist...
Add
Remove
OK
Cancel

```
SELECT
  CLAU.INVENTORY.QUANTITY
FROM
  CLAU.INVENTORY
WHERE
  CLAU.INVENTORY.ITEMID = :itemId
```

```
UPDATE
  CLAU.INVENTORY
SET
  QUANTITY = :quantity
WHERE
  CLAU.INVENTORY.ITEMID = :itemId
```

```
SELECT
  CLAU.INVENTORY.DESCRIPTION,
  CLAU.INVENTORY.QUANTITY,
  CLAU.INVENTORY.ETA
FROM
  CLAU.INVENTORY
WHERE
```

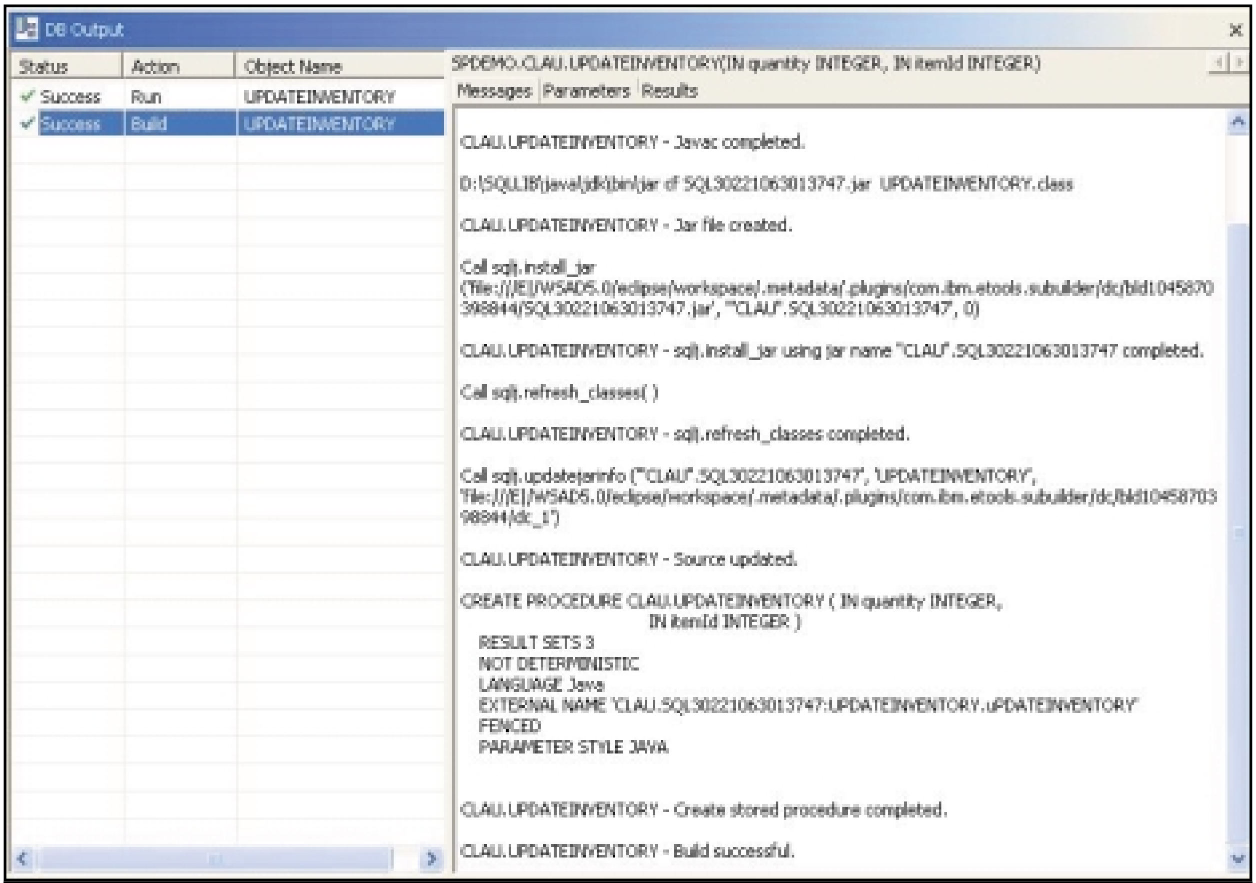
FIG. 2: CREATING A STORED PROCEDURE BY ADDING MULTIPLE SQL STATEMENTS

The DB Output view shows the processes that have been run on the left and contains three tabs on the right: Messages, Parameters, and Results. The Messages tab shows console output returned from the database server. This tab is useful if there is a failure since it often provides an error code or trace of some kind. The Parameters tab shows both input and output parameters and their values. The Results tab shows the result set returned. If there are multiple result sets, you can use the arrow key to page through the multiple result sets.

To run the stored procedure, select the Run action in the pop-up menu for the stored procedure. Figure 4 shows the wizard that will prompt you for the input parameter values. In this example, we entered 12 for the quantity required and 100 for the item ID. The results of the run action are shown in the DB Output view described above.

The Web Services Object Runtime Framework

Now that we have successfully created and tested our stored procedure, it is time to wrap it as a Web service so that it can be invoked over the Internet. To run our stored procedure as part of a Web service, we will need to use the Web Services Object Runtime Framework (WORF). WORF is included in WebSphere Studio and is also available as a separate download. It provides an environment to create XML-based Web services that access DB2. WORF uses SOAP and a Document Access Definition Extension file (DADX) that describes the set of SQL operations that can be invoked over the Internet.



Status	Action	Object Name
✓ Success	Run	UPDATEINVENTORY
✓ Success	Build	UPDATEINVENTORY

```
SPDEMO.CLAU.UPDATEINVENTORY(IN quantity INTEGER, IN itemId INTEGER)

CLAU.UPDATEINVENTORY - Javac completed.

D:\SQLLIB\java\jdk\bin\jar cf SQL30221063013747.jar  UPDATEINVENTORY.class

CLAU.UPDATEINVENTORY - Jar file created.

Call sqlj.install_jar
('file:///E|/WSAD5.0/eclipse/workspace/.metadata/.plugins/com.ibm.etools.subuilder/dc/bld1045870
398844/SQL30221063013747.jar', '"CLAU".SQL30221063013747', 0)

CLAU.UPDATEINVENTORY - sqlj.install_jar using jar name "CLAU".SQL30221063013747 completed.

Call sqlj.refresh_classes( )

CLAU.UPDATEINVENTORY - sqlj.refresh_classes completed.

Call sqlj.updatejarinfo ('"CLAU".SQL30221063013747', 'UPDATEINVENTORY',
'file:///E|/WSAD5.0/eclipse/workspace/.metadata/.plugins/com.ibm.etools.subuilder/dc/bld10458703
98844/dc_1')

CLAU.UPDATEINVENTORY - Source updated.

CREATE PROCEDURE CLAU.UPDATEINVENTORY ( IN quantity INTEGER,
                                        IN itemId INTEGER )
   RESULT SETS 3
   NOT DETERMINISTIC
   LANGUAGE Java
   EXTERNAL NAME 'CLAU.SQL30221063013747:UPDATEINVENTORY.uPDATEINVENTORY'
   FENCED
   PARAMETER STYLE JAVA


CLAU.UPDATEINVENTORY - Create stored procedure completed.

CLAU.UPDATEINVENTORY - Build successful.
```

FIG. 3: THE DB OUTPUT VIEW DISPLAYS THE JAVA BUILD STATUS

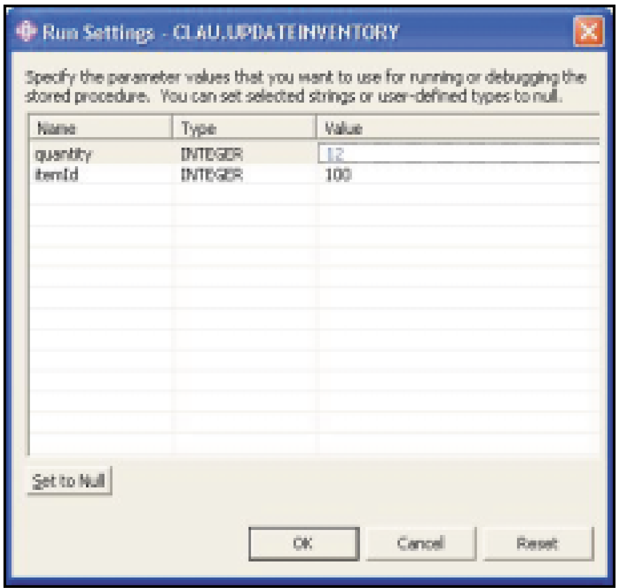
KENETIKS

WWW.KENETIKS.COM

WORF supports both HTTP GET and POST operations in addition to a SOAP request. On a service request, WORF will load the DADX file specified in the request, connect to DB2, run the SQL statement, and commit the database transaction. It will format the result into XML, converting data types as necessary, and return the response to the requester. Optionally, you can also use WORF to generate WSDL, XML Schema, documentation, or a test page.

Creating a DADX Group

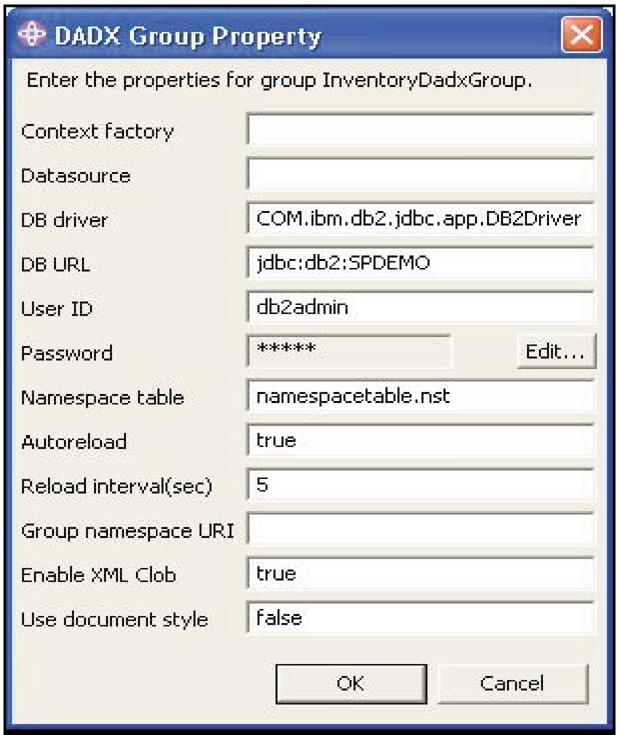
The DADX file used by WORF is deployed as part of a DADX group located in the WEB-INF/classes/groups directory of your Web application. The DADX group contains information, such as database connection parameters, that is shared between DADX files within the DADX group.


Run Settings - CLAU.UPDATEINVENTORY

Specify the parameter values that you want to use for running or debugging the stored procedure. You can set selected strings or user-defined types to null.

Name	Type	Value
quantity	INTEGER	12
itemId	INTEGER	100

FIG. 4: SPECIFYING PARAMETER VALUES FOR RUNNING THE STORED PROCEDURE


DADX Group Property

Enter the properties for group InventoryDadxGroup.

Context factory	
Datasource	
DB driver	COM.ibm.db2.jdbc.app.DB2Driver
DB URL	jdbc:db2:SPDEMO
User ID	db2admin
Password	*****
Namespace table	namespacetable.nst
Autoreload	true
Reload interval(sec)	5
Group namespace URI	
Enable XML Clob	true
Use document style	false

FIG. 5: CREATING A DADX GROUP

To create a DADX group that will hold the Web services that access the database, we use the Web Services DADX Group Configuration wizard shown in Figure 5. We will change the DB URL to jdbc:db2:SPDEMO so that the correct database association is made with the group. The wizard stores the information in the group.properties file in the directory created for this group. The wizard also updates the Web application deployment descriptor, web.xml, to store the appropriate information, such as servlet mapping for this WORF-based Web application.

Generating the DADX File

Now it is time to create the DADX file. WebSphere Studio provides a DADX wizard that you can use to generate the DADX file. Using the DADX wizard, SQL statements or DAD files can be selected to wrap as a Web service. The corresponding DADX operation wrapper is automatically generated by the wizard. Listing 2 shows the DADX file for invoking the UPDATEINVENTORY stored procedure we created earlier.

The <dadx:SQL_call> tag contains the actual call to the stored procedure UPDATEINVENTORY. The <dadx:parameter> tags define the two parameters for the stored procedure. The <dadx:result_set> tags define the two result sets returned by this stored procedure.

The <dadx:result_set_metadata> tags define the result set metadata, including column data types and names. Since the JDBC metadata for a CALL statement does not include the result set metadata, it must be defined explicitly in the DADX file. Our UPDATEINVENTORY stored procedure returns two result sets, and both of them are defined in the Inventory.dadx file.

Creating a Web Service from a DADX File

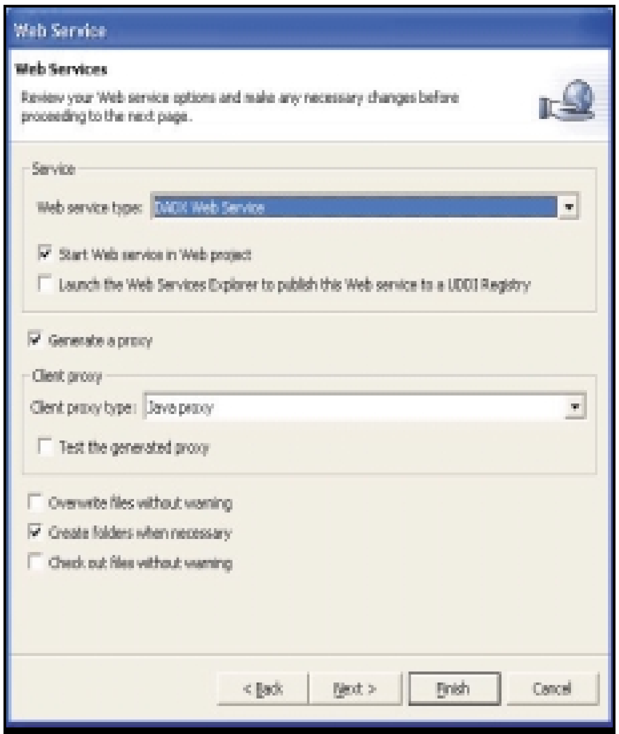
With the DADX group and DADX file in hand, we can now launch the Web Service wizard to create the Web service from the DADX file. Figure 6 shows the Web Service wizard. Note that this is the same wizard used when creating a Web service from a JavaBean or an Enterprise JavaBean. Make sure to select DADX Web Service as the Web service type. In the following page, we specify the Inventory.dadx file created earlier for the stored procedure. We check the “Test the generated proxy” option on the Web Service Test page to generate a set of JSPs that can be used to test the DADX Web service.

Testing the Web Service

Once the generation is completed, the test client shows a list of methods that can be used to test the Web service. We select the UpdateInventory operation. The input pane prompts us to specify the parameters for calling the stored procedure. Once we enter the value for the quantity and the item ID, the stored procedure UPDATEINVENTORY is called, and the result sets are returned in XML format and displayed in the result pane of the test client.

QUEST SOFTWARE

HTTP://JAVA.QUEST.COM/JCSV/WDJ


Web Service

Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

Service

Web service type: DADX Web Service

[✓] Start Web service in Web project

[ ] Launch the Web Services Explorer to publish this Web service to a UDDI Registry

[✓] Generate a proxy

Client proxy

Client proxy type: Java proxy

[ ] Test the generated proxy

[ ] Overwrite files without warning

[✓] Create folders when necessary

[ ] Check out files without warning

FIG. 6: GENERATING A DADX WEB SERVICE

Summary

We have shown you how to use WebSphere Studio to easily create a stored procedure to access and update a database and how to create a Web service from the stored procedure. Using the WORF framework and a DADX file, you can create XML-based Web services that access DB2 data and stored procedures using a call operation in the DADX file to call a stored procedure. You can also specify operations such as update or query to invoke SQL statements directly to insert, update, delete, or query your data. Operations such as storeXML and retrieveXML can also be included in the DADX file if you want to use the DB2 XML Extender to store or retrieve XML documents to or from your database.

References

- *Web Services Object Runtime Framework (WORF) for DB2*: www7b.software.ibm.com/dmdd/zones/webservices/worf
- *IBM WebSphere Application Server - Express:* www-3.ibm.com/software/webservers/appserv/express/features.html
- *IBM WebSphere Studio Application Developer:* www-3.ibm.com/software/ad/studioappdev
- *DB2 Developer Domain:* www7b.software.ibm.com/dmdd

LISTING 1: THE JAVA STORED PROCEDURE

```
import java.sql.*; // JDBC classes

public class UPDATEINVENTORY {
  public static void UPDATEINVENTORY(int quantity.int
  itemId,
        ResultSet throws SQLException
  {
        // Get connection to the database
        Connection con = DriverManager.
        getConnection("jdbc:default:connection"

        String sql = "SELECT CLAU.INVENTORY.QUANTITY"
                + " FROM CLAU.INVENTORY"
                + " WHERE CLAU.INVENTORY.ITEMID = ?";
        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt(1, itemId);
        stmt.execute();

        ResultSet rs = stmt.getResultSet();
        rs.next();
        int currentQuantity = rs.getInt(1);

        if (currentQuantity > quantity)
        {
                int newQuantity = currentQuantity -
                quantity;
                sql = "UPDATE CLAU.INVENTORY SET QUANTI
                TY = ? "
                        + " WHERE CLAU.INVENTORY.ITEMID
                        = ?";
                stmt = con.prepareStatement(sql);
                stmt.setInt(1, newQuantity);
                stmt.setInt(2, itemId);
                stmt.executeUpdate();

                sql = "SELECT CLAU.INVENTORY.DESCRIPTION,
                CLAU.INVENTORY.QUANTIY, C
                        + " FROM CLAU.INVENTORY"
                        + " WHERE CLAU.INVENTORY.ITEMID
                        = ?";
                stmt = con.prepareStatement(sql);
                stmt.setInt(1, itemId);
                stmt.execute();
                resultSet[0] = stmt.getResultSet();
        }
  }
}
```

LISTING 2: INVENTORY.DADX FILE

```
<?xml version="1.0" encoding="UTF-8"?>
<dadx:DADX xmlns:dadx="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xsi:schemaLocation="http://schemas.ibm.com/db2/dxx/dadx
      dadx.xsd">

    <dadx:result_set_metadata rowName="result" name="detail">
        <dadx:column name="DESCRIPTION" type="VARCHAR" nul
        lable="true" />
        <dadx:column name="QUANTITY" type="INTEGER" nul
        lable="true"/>
        <dadx:column name="ETA" type="VARCHAR"
        nullable="true"/>
    </dadx:result_set_metadata>

    <dadx:result_set_metadata rowName="result" name="ignore">
        <dadx:column name="QUANTITY" type="INTEGER" nul
        lable="true"/>
     </dadx:result_set_metadata>

    <dadx:operation name="UpdateInventory">
        <dadx:call>
            <dadx:SQL_call>CALL UPDATEINVENTORY(:quantity,
            :itemId)</dadx:SQL_call>
            <dadx:parameter name="quantity" type="xsd:int"
            kind="in" />
            <dadx:parameter name="itemId" type="xsd:int"
            kind="in" />
            <dadx:result_set name="results1"
            metadata="ignore" />
            <dadx:result_set name="results2"
            metadata="detail" />
        </dadx:call>
    </dadx:operation>
</dadx:DADX>
```

MACROMEDIA

WWW.MACROMEDIA.COM/GO/WSDJ/

API provides full support for the EJB 2.0 specification

Data Access Beans Using IBM's Extended API in WebSphere Application Server 5.0

BY SOUTIK SINGHA, DEEPENDRA SHRESTHA, AND RAGHU SHRESTHA,

IBM WebSphere Application Server provides a really useful API to access data from back-end databases using data access beans. Applications can access data from back-end databases using the JDBC API, the standard J2EE-defined application programming interface. But in our programming experience we've seen that the standard APIs don't always provide a complete solution. And in the past we've had to write our own frameworks on the standard APIs to make it easier for developers to code.

ABOUT THE AUTHOR
Soutik Singha is a senior enterprise architect and a certified systems expert at Noospherics Technologies. He has 10 years of experience in IT, working in several industries and in roles including mentor, tech lead, and architect. Soutik has written a number of articles on aspects of WebSphere.

E-MAIL
soutik@yahoo.com

With WebSphere Application Server 5.0 IBM has provided such a framework, with a rich set of features such as parameterized queries, the ability to use caching for result sets and to update through the cache, and metadata mapping support, to name a few. At the same time it hides much of the complexity associated with data access from back-end databases. This API is basically an extension of the JDBC API. Also, the data access beans help to conform to the JDBC 2.0 RowSet standard. At the same time, this API minimizes the gap between the JDBC (Java Database Connectivity) and the JCA (J2EE Connector Architecture) standards, providing not only full support for the EJB 2.0 specification, but also capabilities like entity bean inheritance, access intent policies, the ability to cache data across transactions, and several other features that go beyond the specification. Some key advantages of using this API include:

- Flexibility
- Ease of use
- More control over the operations and the results retrieved
- The ability to reuse resources
- Support for all kinds of SQL operations
- No specificity to particular database

This API isn't new. In the days of VisualAge for Java some of us used the com.ibm.db package in the data access applications that were shipped within the ivjdab.jar. IBM now ships dbbeans.jar with WebSphere Studio Application Developer and WebSphere Studio Site Developer. This JAR contains a new package called com.ibm.db.beans. Although these products still support the use of the earlier package for existing applications, IBM strongly suggests using dbbeans.jar instead for new applications. In Studio, the Data Access Bean API is packaged in the dbbeans.jar file, which is located in the <WSAD_ INSTALL_DIR>\wstools\plugins\com.ibm.etools.dbjars_5.0.1\jars folder. You can also see the JavaDocs API in the same folder. Adding the JAR file to the project build path allows you to use the API in your code. It is not possible within the scope of this article to show all the features of this API, but we will try to show a few that are very useful to developers.

Connecting to a Database

Let's consider a connection object used to connect to a back-end database. The API supports both shareable and unshareable connections. We call a connection unshareable when it is not shared with other components and is fully controlled by the component using it. But within the scope of a transaction a shareable connection may be shared by multiple components. This can be achieved through multiple connection handles instead of creating a new physical connection every time. This is possible only when the well-known getConnection method returns the same connection with the same properties. WebSphere caches a connection handle and holds it across transaction and method boundaries. This facilitates more efficient use of resources.

The JCA specification supports this kind of sharing by setting the required properties during the getConnection call to the resource adapter. A ConnectionSpec object containing all the necessary connection properties is used for this purpose. But there is no such interface in the JDBC API. The IBM extended API has a DBConnectionSpec class in the com.ibm.db.beans package for this purpose. This class contains only the connection properties that should not be changed after a connection has been obtained. This facilitates the reusability feature of shareable connections. Also, if a DataSource property is used, DBConnectionSpec eliminates the costly JNDI lookup required each time.

SQL Operations

The API also provides an abstract class, called DBStatement, that is the super class of other important classes like

ERROR CODES	DESCRIPTIONS
badUidPwd	Trying to connect when username and password supplied is not correct
cacheEmpty	Trying to get data when the result set cache is empty
cannotConvertToString	If a column value cannot be converted to a string type
connectionClosed	Trying to use a connection that is already closed
dbbaseDriverNotFound	If the JDBC DriverManager cannot find the specified driver
dbNotExecuted	If the SQL statement has not been executed
inconsistentColumnName	If the metadata contains a different column name from the actual column name in the database
invalidDirection	If the fetch direction from the cache is not one of FETCH_FORWARD or FETCH_REVERSE
lockNotSupported	If the method lockRow is not supported by the database
multipleTables	Trying to insert, update, or delete in a result set obtained through a join, i.e., from multiple tables
noColumnUpdate	Trying to set a value in a column that is not updatable
noInitialContext	If the initial context cannot be created while trying to connect to the database
noSQL	When the command property of a DBStatement contains a null or empty string and there is an attempt to execute the statement
notOpen	If the result set is not open
noTableDefined	If no such table is defined in the database upon which a statement is attempting some SQL operations
noTransactions	If auto commit false is not supported by the database
parameterNotDefined	If a parameter in the specified position is not defined in a statement
readOnly	If the result set is read-only and an attempt is made to change values in it
rowChanged	If the current row of the result set cannot be modified or deleted because there is no such row in database
rowNotFound	If a lock on the current row cannot be obtained because there is no such row in the database
transactionIsolationError	If the specified transaction isolation level cannot be set
wrongObjectType	If the value supplied to set a column or parameter in a statement is different from the Java type defined in the metadata.

TABLE 1: SOME DBEXCEPTION ERROR CODES

DBSelect and DBModify. The idea behind DBStatement is to expose the JDBC 2.0 RowSet functionality. A RowSet not only wraps up a connection, a statement, and the result set as a bean but also provides methods and properties to use them. DBSelect can be used to execute a query, and it is possible to insert, update, and delete rows in the result set returned without writing any more statements. The setCommand() method should be used to set the SQL query associated with the statement. There are provisions for using parameterized queries to get, set, and find parameters in the underlying SQL statement. The API also provides the ability to cache the result set obtained and navigate within the result set in any direction, including jumping directly to a particular row. Listing 1 shows the use of DBSelect.

DBSelect's setCacheLimit(int) sets the maximum number of rows from the result set to be put into the cache.

```
dbselect.setCacheLimit(10);
// put 10 rows from the
result set in cache
dbselect.execute();
```

To retrieve results from the cache, use getCacheLimit(), which returns an int value corresponding to the cache limit set. This can be used to traverse through the cached resultset. The getCacheValueAt(rowIndex, colIndex) method gets the value of column specified by colIndex for the row indicated by rowIndex.

```
dbselect.getCacheValueAt(1,
1);
// retrieves the value of
column 1 at row 1
```

DBModify is used to execute statements that do not return result sets. Although the result set dealing with properties of a RowSet cannot be used, other properties, such as setting connection properties and setting parameters, are supported (see Listing 2).

The number of affected rows can be determined by using the getUpdateCount() method:

```
dbmodify.getUpdateCount();
// returns an int value rep-
resenting the number of
updated rows.
```

ABOUT THE AUTHOR

Deependra Shrestha is a Sun Certified Java programmer and IBM Certified WebSphere developer with extensive experience in analysis, design, development, and administration of WebSphere and J2EE applications. He is a senior programmer for Client Network Services, Inc. Deependra has been involved in a number of WebSphere and J2EE projects for various large organizations.

E-MAIL
deependra202@yahoo.com

DBProcedureCall extends DBSelect and allows the calling of a stored procedure from the back-end database and other functionalities to manage stored procedure results. You can call the DBStatement.setConnectionSpec() to set the connection spec on a statement. In this way more than one statement can use the same connection spec. The DBParameterMetaData class allows you to get and set metadata for parameters for different DBStatement classes, e.g., in order to execute a parameterized SQL call using DBSelect or DBModify or a stored procedure using DBProcedureCall. It can be retrieved using the getParameterMetaData() method of DBStatement. To be passed or retrieved from the call the parameters should be properly defined and set before the call is executed. The parameters can be set to a custom Java class or to the standard data types. This is useful for mapping the Java types to the database fields to get and set values before actually executing a statement (see Listing 3). Then execute the query with the following command:

```
dbmodify.execute();
```

The DBEvent class extends the EventObject from java.util that marks different events associated with database operations. The API also provides two event listener interfaces, DBAfterListener and DBBeforeListener, that extend the java.util.EventListener interface. These listeners can be used to identify actions affecting the RowSet or the database both before and after the events concerned.

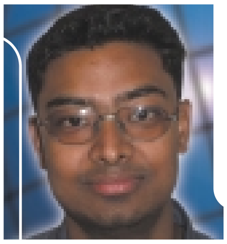
Exception Handling

There is only one exception class, called DBException, that is defined in the com.ibm.db.beans package. This class extends java.sql.SQLException and has many features that can be used to identify the source of problems. DBException defines more than 70 different error codes. Some of the most useful ones, along with their descriptions, are shown in Table 1. Please check the API documentation for a complete list of error codes.

WebSphere also supplies two exceptions that are not part of the com.ibm.db.beans package but that can be used by developers to check the state of a connection. They are com.ibm.websphere.ce.cm.ConnectionWaitTimeoutException, used to check if the connection has timed out; and com.ibm.websphere.ce.cm.StaleConnectionException, used to check stale connection problems.

Conclusion

The J2EE specification goes only so far in getting vendors to adhere to a set of rules. It's the job of application server vendors to differentiate their products from others. In an effort to do so they add some very valuable frameworks and packages that developers should make use of to do away with tedious coding that achieves the same results. The packages we discussed here extend the JDBC API's functionality and make it easier to code to JCA standards. The advantages inherent in using the extensions are important to all WebSphere developers who want to go that extra mile without the hassle of discovering and implementing all the intricate details.



ABOUT THE AUTHOR

Raghu Shrestha is a Sun Certified J2EE Programmer and an IBM Certified WebSphere developer with comprehensive experience in application server technologies. He worked extensively as a WebSphere v5 release testing specialist. He has extensive analysis, design, development, and administration skills in these technologies.

E-MAIL
raghu@noospherics.com

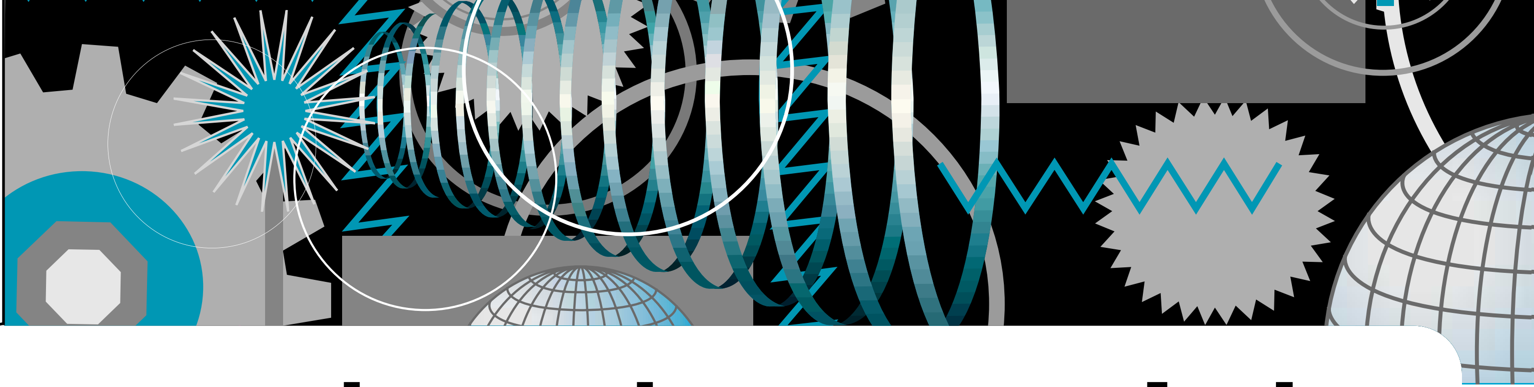
LISTING 1:
```
// Importing the required package
import com.ibm.db.beans.*;

//Instantiate the DBSelect object
DBSelect dbselect = new DBSelect();

// DBSelect object is used to set the Database connection
// information

dbselect.setDriverName("COM.ibm.db2.jdbc.app.DB2Driver");
//Change the driver name for other databases

dbselect.setUrl("jdbc:db2:TEST");
dbselect.setUsername("administrator");
dbselect.setPassword("admin");

// the setCommand method is used to set the SQL being executed
dbselect.setCommand("SELECT * FROM USERINFO");

// executing the SQL statement
dbselect.execute();

boolean hasRecords=dbselect.onRow();
if(hasRecords)
{
do{
// getColumnAsString retrieves the data as String
// from the specified column index.
   String strFirstColumn=dbselect.getColumnAsString(1));
   String strSecondColumn=dbselect.getColumnAsString(2));
   String strSecondColumn=dbselect.getColumnAsString(3));

}while(dbselect.next());
}
//the close() method will free up JDBC resources and closes
the //connection
 dbselect.close();
```

LISTING 2:
```
DBModify dbmodify = new DBModify();
dbmodify.setDriverName(“COM.ibm.db2.jdbc.app.DB2Driver”);
dbmodify.setUrl(“jdbc:db2:TEST”);
dbmodify.setUserName(“administrator”);
dbmodify.setPassword(“admin”);
// Setting the command string:
String insert = “INSERT INTO UserInfo (FirstName, LastName,
Address, Telephone) VALUES ('Bob', 'Lange', 'Maryland',
'1234567890')”;

// or String update = “UPDATE USERINFO SET TelNum =
'222222222' //                                          WHERE
FirstName='Bob'";

dbmodify.setCommand(insert);
// or dbmodify.setCommand(update);

dbmodify.execute();
// execute the insert statement
```

LISTING 3:
```
dbmodify.setCommand("UPDATE USERINFO SET Telephone = ? WHERE
FirstName='Bob'");

// get the parameter meta data for the DBModify object.
DBParameterMetaData dbmetadata =
dbmodify.getParameterMetaData();

// describe the parameter, the int 1 defines the index of
the //parameter.
dbmetadata.setParameterName(1,"tel");
dbmetadata.setParameterType(1,java.sql.Types.VARCHAR);

// set the parameter
dbmodify.setParameter("tel","1110004567");
```

DIRIG SOFTWARE

WWW.DIRIG.COM/CORP/WP_REQUEST.HTML



WebSphere Risk Management Part 1

Monitoring application simplifies the task

– BY BASSEM W. JAMALEDDINE –

ABOUT THE AUTHOR
Bassem W. Jamaleddine is a Web systems engineer who has orchestrated the development of several projects at IBM's T.J. Watson Research Center, including IBM's Java-based network computer and the new generation of WebSphere Application Server technology. Bassem is the author of *IBM WebSphere Application Server Programming* (McGraw-Hill).

E-MAIL
bassem@tcnd.com

In this article, the first part of a two-part series, I will present WASLED/WASMON, a WebSphere monitoring application, and show you how you can use it to monitor WebSphere Application Server and to plan a WebSphere risk management procedure.

I will discuss some of the system resources that need to be made available to ensure the operability of WebSphere Application Server. You will learn how to configure WASMON to ensure that these resources are available and up and running. I will show you how to prepare the essential recovery scripts and make them available within the WASMON repository so you can trigger specific actions upon failure notification. You will then learn how to communicate with WASMON via the Internet to initialize an administrative action.

In this article I use the example Web application WASDG, which was used throughout my book, *IBM WebSphere Application Server Programming*. In addition, for brevity and generic notation of WAS commands, I will use the WASDG environment notation (also defined in my book). For example, $WASLOG_STDOUT refers to the fully specified name of the file to which WAS writes the standard output messages; $SEAPPINSTALL refers to SEAppInstall, $WASSTOP; and $WASSTART refers to the combined command to restart WAS. The test environment for the risk management scenario consisted of Pentium III, 750MHz, 2GB RAM, Linux servers running Red Hat v7. The applications used are WebSphere Application Server Single Server Edition v4, downloadable from www.ibm.com; and WASLED/WASMON v1.2.2, downloadable from www.tcnd.com.

Defining/Asserting the Operability of a WebSphere Region

A WebSphere region encompasses many server machines running processes to render the functionality of a WebSphere domain and its Web applications. A WebSphere region therefore consists of servers that are running WAS processes, an HTTP server, one or more database servers, and a monitoring application such as WASLED/WASMON. WAS processes are typically running to fulfill any of the following:

- To store and retrieve information from the WAS configuration repository
- To manage and serve requests redirected by the HTTP server
- To service or to organize workload distribution
- To supervise WAS containment and cache cleanup (for the Web container and the EJB container)

A WebSphere region is termed valid and operational when all the necessary resources and services hosted by the server machines are in place and able to serve the Web applications. Consider the WebSphere region depicted in Figure 1.

In this figure, the Internet server hserv.tcnd.com is an HTTP server that is patched with WAS's vendor plug-in to forward requests to the WAS server at node1.tcnd.com. The session persistence database and the DataSource are managed by the UDB server installed on db.tcnd.com. Let's take

a look at how you can use WASMON to monitor the servers and resources in a WebSphere region.

The WASMON Application

WASLED/WASMON v1.2.2 can be downloaded from www.tcnd.com. For the remainder of this article I will refer to WASLED/WASMON simply as WASMON, unless it is necessary to make a distinction between the two.

WASMON consists of two main programs, wasmon and wasmonhelper, along with miscellaneous programs such as wasmoncl, the client that connects to wasmon; wasmontkt, the ticket and configuration checker; and wasmonvar, the directive-to-variable mapper.

WASMON monitors WAS without using any WebSphere APIs. The application permits monitoring of the WAS runtime or of a Web application runtime simply by preprocessing the WAS log file or the Web application log file. Such monitoring is therefore dependent on the logging of information to a file system. WASMON also offers a supervisor that allows you to monitor a WebSphere region independently of the log files. The WASMON supervisor mode evolved to ensure that a WebSphere region is operational whenever data ceases to flow (or no more exceptions are thrown) from WAS components or the hosted Web applications.

The WASLED/WASMON Console

By default Linux installations have the prerequisites needed to run WASMON, except for Perl/Tk, which needs to be installed manually. To start the WASLED/WASMON console, execute the following command:

```
# wasmon
```

Figure 2 shows the WASLED/WASMON console in its active state. The console consists of three panes.

- In the first pane you can make entries using the keyboard or the mouse to interact with WASMON. In addition, this pane has two throbs that show the varying state of the WAS Web container and the WAS EJB container.
- The second pane, also called the WASLED pane, is a graphical representation of the activities of the WAS components. When a component is detected by WASMON, it is dynamically allocated a graphical object to show its current state. Each graphical object is shown on one line, as shown in Figure 3. The line is formed of three parts: the component's descriptive name, the last LED component processed by WASMON (its color reflects the state of the LED previously processed), and a colored bar that shows the status of the current processed component.
- The third pane is a log showing the wasmon program's activities.

The console shown in Figure 2 has been set to active and is monitoring WAS runtime components. To activate WASMON, enter a socket number in the upper corner of the console and click on Listen; this starts WASMON as a server to process WAS logging records on socket 12345. Then start wasmoncl on a workstation that has access to the WAS log files. wasmoncl will connect to WASMON listening on socket 12345 on supervise.tcnd.com and send the WAS log information to be processed. For example, on node1.tcnd.com we will start the command:

```
# tail -f  $WASLOG_STDOUT | perl wasmoncl
12345 supervise.tcnd.com
```

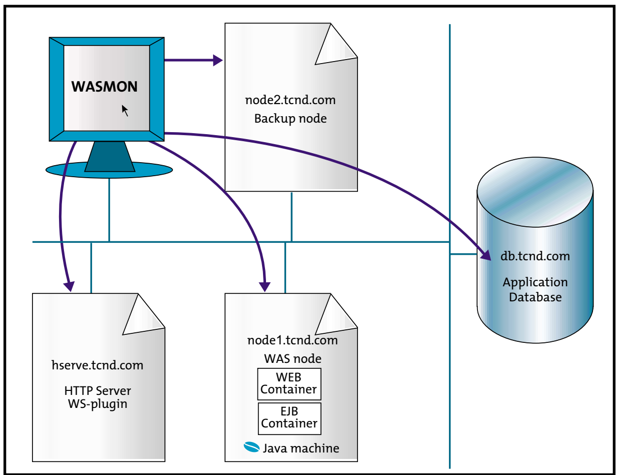


FIG. 1: A TYPICAL WEBSPHERE REGION SHOWING THE ESSENTIAL SERVERS INVOLVED IN RENDERING IT

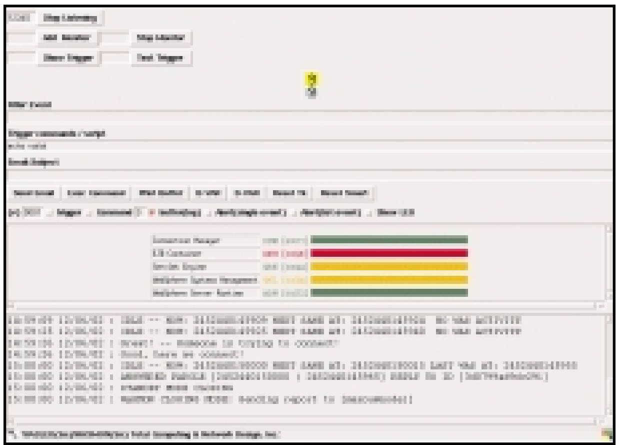


FIG. 2: THE WASMON CONSOLE IN ITS ACTIVE STATE

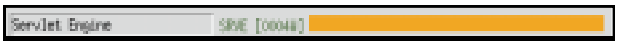


FIG. 3: SERVLET ENGINE COMPONENT (SRVE) IN WASLED PANE

Finally, let's take a look at a very simple example to quickly test WASMON. When WAS is restarted, the SRVE component (that corresponds to the WAS servlet engine) will be active, and typically an SRVE0091I is logged to the file. For this reason, to filter an event while WASMON is running, you can just enter the string to be matched in the Filter Event text box; for instance, you could enter the regular expression SRVE00[0-9][0-9]I. You will also specify a trigger to associate with the filter; for example, specify trigger 0002 and check the trigger button (see Figure 4). Remember to make WASMON active by entering a socket number and clicking on Listen.

Now restart WAS (as we did on node1.tcnd.com). As WAS is restarting, SRVE0091I is detected by WASMON and the associated trigger 0002 is executed to simply print "triggering script wm_0002" to the terminal where WASMON was started. Trigger 0002 corresponds to the shell script ./TriggerPool/wm_0002.sh:

```
#!/bin/ksh
echo triggering script ws_0002
```

You can also consider testing with trigger 0003, which corresponds to the shell script ./TriggerPool/wm_0003-notify-all-users.sh:

```
#!/bin/ksh
wall WASMON notifying all users, and trig-
gering 0003
```

Both of these scripts are simple scripts used for demonstration. Later in this article I will show you more interesting scripts that can initiate the takeover of a WAS server by another.

The second pane (WASLED pane) of the console shows the LEDs of the many WAS components that are detected by WASMON. In our case, SRVE0091I should be shown in the WASLED pane. In the first pane, the top throb should have changed state because the WAS Web container is active.

In the previous example, I showed you the simplest way to detect and filter WAS events. However, WASMON reads its configuration from wasmon.conf, and manually specifying a regular expression to be filtered and a trigger number to be executed is only useful for online monitoring. The following directive, in wasmon.conf, shows a more interesting way to initialize WASMON monitoring for the WAS Web container component (SRVE):

```
ALERT ON COMPONENT LIST WARNS:
1,1,0002,,admin@tcnd.com,,WARNING ON SRVE
and CNTR,,,SRVE,CNTR
```

What follows the directive shown above is a list of fields based on the following template:

FIG. 4: WASMON MONITORING THE WASDG WEB APPLICATION

```
number of times to send email, number of
times to trigger the script, 4-digit trig-
ger number, an argument to be passed to the
trigger, e-mail address, cc-email-address,
email-subject, email-body, logical-expres-
sion, filter1[,filter2[...][,filterN]]
```

The directive states: "Filter any warning thrown by the WAS runtime components SRVE (Servlet Container) and CNTR (EJB Container), if an exception is thrown by any of these two components, then send (only once) an e-mail to admin@tcnd.com with the subject 'WARNING ON SRVE and CNTR,' and trigger (only once) the shell script that is bound to the trigger number 0002."

Let's take a look at two other variations of the directives. The first is to monitor the WAS Web container using a regular expression as a filter:

```
ALERT ON FILTER:
1,1,0002,,admin@tcnd.com,,Sending alert from
WASMON,,,SRVE[0-9][0-9][0-9][0-9][A-Z]
```

The second directive is to monitor WAS using a logical expression that asserts that "either the WAS Web container OR the EJB container is in error, AND WAS is running low on memory."

```
LOGICAL ALERT ON LIST FILTERS:
0,1,0002,,admin@tcnd.com,,Alert WAS low on
memory, ,(c1 || c1) && c3,SRVE[0-9][0-9][0-
9][0-9]E,CNTR[0-9][0-9][0-9][0-9]E,running
low on memory(.)*destroyed
```

This second monitoring directive requires the evaluation of the following logical expression: (c1 || c2) && c3; where c1 is the filter for the Web container, c2 is the filter for the EJB container, and c3 is the filter for a string stating that WAS is low on memory.

The expression at the end of the above directive is a regular expression where (.)* means "match anything that falls between the word 'memory' and the word 'destroyed..'"

Because it is possible to specify regular expressions as filters, the use of WASMON monitoring can also be applied to any log file that you wish to filter specific data from. In this case, you need to use the filtering directives in which you can specify regular expressions: ALERT ON LIST FILTERS and its logical counterpart (that can accept a regular expression) LOGICAL ALERT ON LIST FILTERS.

By default WASMON is set to monitor all 40 WAS components. You can remove a component or add another component to WASMON simply by typing its four-letter acronym in the first pane and clicking on Add Monitor or Stop Monitor, respectively.

WASMON monitoring can also be applied to Web applications that throw exceptions similar to the way WAS throws its exceptions. *IBM WebSphere Application Server Programming* shows a full-blown exception handler (called the WasdgException) that throws exceptions (and LEDs) similar to the way WAS does. The exception handler uses messages bundled using the BundleManager. A Web application that uses the WasdgException handler can also be monitored using WASMON; in this situation, you would start WASMON as follows:

```
# wasmon -xwasled -wc <Web_app_components> -
wl <Web_app_leds>
```

I am not going to elaborate on Web application monitoring, but I mention it here because of its importance. The above command starts WASMON by excluding the monitoring of all WAS components and including the components listed in Web_app_components file, and by reading the LED mapping from Web_app_leds file.

So far we have seen that WASMON can filter the data as it is being logged by the WAS runtime or the Web application runtime. But a true monitoring application should not depend on the runtime of the application server being monitored. There might be situations in which WAS is dead, or the JNDI name lookup is not responding because the WAS EJB container is dead (or idle due to a race condition on threads trying to clean up the cache). WASMON monitoring should not depend exclusively on the messages logged by WAS or its Web applications.

Therefore, WASMON monitoring is based on filters that can be mapped to logical expressions. WASMON logical expressions are formed from conditional variables, also called monitoring or diagnostic variables. These variables are set as a result of filtering WAS log data, or by WASMON after scrutinization of the systems and resources in a WebSphere region. This last capability of WASMON monitoring (carried out independently of WAS) is based on a set of Boolean variables called b-var, and a set of differential variables called d-var. The b-var and d-var variables are set by a WASMON delegator program called wasmonhelper.

We will start our demonstration by writing the configuration file, wasmonhelper.conf, which directs wasmonhelper to collect data about the WebSphere region.

What we are trying to accomplish is a server takeover initiated by WASMON. For example, consider Figure 5, in which node2.tcnd.com is to replace node1.tcnd.com.

WASMON makes the replacement of a failing server with a back-up server possible. It is necessary to follow a strategy based on detection, assertion, and decision – the "big three" functions of WebSphere risk management.

- ***Detection:*** The detection of the a malfunctioning WAS component or system resource can be handled by WASMON through the use of filtering and associated monitoring variables.
- ***Assertion:*** The detected variables can be combined into a logical expression that is evaluated by WASMON to assert a certain condition. For instance, what if the WAS Java Virtual Machine is running but not responding to the JNDI name lookup on iiop://was.host.name on port 900? WASMON logical expression will assert such a condition.
- ***Decision:*** If such an error occurs, what should you do? What is the next step? In WASMON the action is initiated by one or more scripts. We will consider such shell scripts that automatically rebuild a J2EE Web application without the attendance of an operator.

Monitoring the WebSphere Region with wasmonhelper

Figure 1 illustrates the systems in the WebSphere region that we wish to monitor. To assert the proper operability of a WebSphere region, let's examine the figure and tabulate some of the essential elements that need to be monitored. Table 1 details usage of the ports, processes, and URIs for each server shown in Figure 1.

We will use wasmonhelper to monitor the systems and resources shown in Table 1. To do so, we will write the configuration file, wasmonhelper.conf, shown in Listing 1. (All of the listings for this article are available at www.sys-con.com/websphere/sourcec.cfm.)

wasmonhelper.conf contains several directives to set up monitor variables. Each of the directives that start with BOOL or DVAR in wasmonhelper.conf maps to a monitor variable whose name can be used in wasmon.conf. To view what a variable will map to, just issue the command wasmonhelper with option ideal. For instance, to dump the b-var and d-var as they are initially set (or ideally as set by wasmonhelper before the program starts to evaluate them), issue the command:

```
# wasmonhelper ideal
```

Listing 2 shows the mapping of the directives shown in Listing 1 to their variables counterparts.

Each variable can be used in the logical expression wasmon.conf by adding the @ symbol as a suffix. In Part 2 of this series I will provide an example.

Your understanding of the usage of wasmonhelper to glance at the system resources defined in a WebSphere region will become clearer when I discuss the wglance command. Now let's clarify the evaluation of the d-var shown on line 18 of Listing 2.

Scrutinizing System Resources: %Memory and %CPU

The wasmonhelper also offers a set of d-var variables that are used to scrutinize the system resource consumption per process. Consider the directive DVAR RSHSCRUTPROCESS. This directive can be used to raise and set a d-var to an escalating positive integer (therefore evaluating to true in a logical condition) when the system consumption of a process reaches a certain limit.

For instance, say WAS is started on the Linux host node1.tcnd.com with process number 32184. To monitor the percentage of memory consumption of this process and to raise an alert when WAS process 32184 consumes more than 8.1 percent of the system memory, we will use the following directive in wasmonhelper.conf:

```
DVAR RSHSCRUTPROCESS: (3) 12345 MEM 8.1
```

The first argument, (3), is a list of values that are reserved and used internally by WASMON. This is called a freeze value and is normally set to 0 or 1 so that wasmonhelper will stop

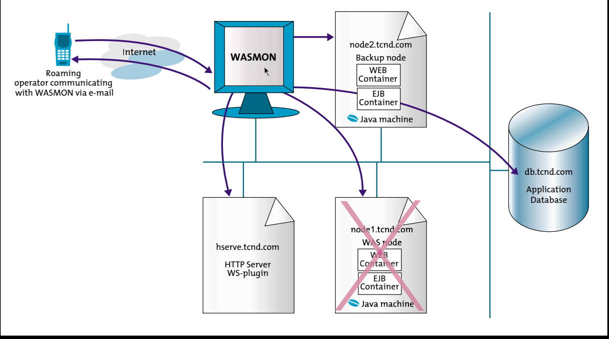


FIG. 5: REPLACING THE FAILING WAS NODE1.TCND.COM BY THE BACKUP WAS NODE2.TCND.COM

(freeze) differentiating the first time the process consumption reaches 8.1% or more of system memory. We set it to 3 so that wasmonhelper will continue differentiating three more times. MEM is the attribute of the process that we seek to scrutinize; it is followed by the number, 8.1, which represents the limit we wish to place on the amount of memory consumed.

Server	Usage
node1.tcnd.com	JNDI name lookup on port 900
node1.tcnd.com	Internal transport port 9080 used for redirection
node1.tcnd.com	Admin port 9090
node1.tcnd.com	URI to administer WAS using port 9090 and URI /admin
node1.tcnd.com	WAS process number as read from the kernel table, e.g., 32184 in my case study
node1.tcnd.com	DumpEnv servlet, defined in WAS Web container, can be requested on the internal port 9080 with URI /wasbook/dumpenv
hserv.tcnd.com	DumpEnv servlet requested through HTTP server redirection.Default port 80 and URI /wasbook/dumpenv
node1.tcnd.com	Deployment descriptor of the webapp defined in the WASDG application: $INSTALLEDAPPS/wasdg.ear/webapp/WEB-INF/web.xml
node1.tcnd.com	ibm-web-ext.xmi defining IBM Web application extension attributes, such as serveServlets ByClassnameEnabled
db.tcnd.com	Connection port 50000 forUDB, also defined in /etc/services
hserv.tcnd.com	HTTP server default port 80
hserv.tcnd.com	Location of the default WS-plug-in plugin-cfg.xml file, e.g., /opt/WebSphere/AppServer/config/plugin-cfg.xml on Linux or check httpd.conf
hserv.tcnd.com	File /etc/hosts contains a definition to host node2.tcnd.com

TABLE 1: HOST NAMES AND SOME OF THE ESSENTIAL RESOURCES TO BE MONITORED

MEM is not the only attribute that you can examine; in fact you can monitor any of the following attributes when using Linux: TRS, DRS, RSS, and MEM; or any of these when using AIX: PGIN, SIZE, RSS, LIM, TSIZ, TRS, CPU, and MEM. Getting a list of the process attributes is fairly simple; just issue the command ps v against the process number on your system.

WASMON monitoring does not use any Java or WebSphere APIs that can heavily impact and degrade the performance of the system being monitored. This essential characteristic of WASMON sets this application apart from libraries or applications offered by IBM. There are myriad reasons why a monitoring application should not depend on the throwable exceptions generated during the runtime of the J2EE application server.

On the one hand, exceptions are typically placed around a segment of code to be evaluated when we suspect a warning or an error, yet this would suggest knowledge beforehand of where an error will occur. On the other hand, an application used for risk management should not just interrogate and give reasons why there is a failure. There is no time to make any reasonable judgment. A risk management application should promote system recovery or initiate a server takeover. For instance, if the JNDI name cannot be looked up, an automatic compilation, rebuild, and deployment of the EJB module should be initiated.

To see the effect of the directive,

```
DVAR RSHSCRUTPROCESS: (3) 12345 MEM 8.1
```

and how the d-var can be used to escalate memory consumption, you need to stress-test the application server and run wasmonhelper in verbose mode. To stress the application server you can use SharkUrl, discussed in Chapter 22 of my book. SharkUrl is freely distributed with the Gramercy ToolKit for noncommercial use. Refer to the article www.tcnd.com/article/p9 for a complete scenario for monitoring the escalation of memory consumption.

So far so good! Now that you know how to get and scrutinize the WAS process and other system resources using wasmonhelper's monitoring variables, all that is left is to show you how to bind such monitoring variables to active scripts with WASMON. However, before discussing how to associate variables and scripts, we need to discuss a few shell scripts that can compile and deploy a Web application automatically. Such scripts will be discussed in Part 2 of this article. Now let's take a look at a command that can fit nicely into the system of any WebSphere developer or administrator: the wglance command.

Glancing at a WebSphere Region: wglance

Although wasmonhelper is used to set diagnostic variables, it is also used generically as a system command to assess the serviceability of the systems and resources in a WebSphere region. Once the product is installed, a system administrator can use wasmonhelper and its counterpart wglance as commands run from the command prompt to glance at and assess the functionality of the many computer systems in a WebSphere region.

A programmer can write a configuration file, wasmonhelper.conf, in his or her $HOME directory, or instead set a configuration file globally for all users in the WASMON installation directory. Consequently, any user can then run wasmonhelper or wglance to glance at WAS and its EJB container service for the JNDI name lookup, or to monitor other resources such as the HTTP server daemon and the UDB active port. Programmers will find these commands handy; wglance is a simple command that will also release the system administrator from trivial questions about the operability of the systems in a WebSphere development environment. The following two commands are equivalent:

```
# wasmonhelper glance
```

and

```
# wglance
```

When wglance is executed without any option, it prints all data that has been evaluated by wasmonhelper. You can follow wglance with a word to print specific data. For instance to glance at the JNDI names lookup:

```
# wglance jndi
```

With this command you can also specify an iteration value and a delay value (to be used during the iteration). For instance, to monitor the memory consumption while scrutinizing the WAS process (using the d-var as shown earlier), we will issue the following command:

```
# wglance MEM  5  d12
```

Also consider the following command, which you can use during the deployment of an EJB module:

```
# wglance DataAccessComponent 10 d14
```

This command will iterate 10 times, pausing 14 seconds after each iteration, and it will print statistics about the resolvability of the JNDI name lookup, DataAccess-Component. Such a command will help programmers demystify a strange situation such as the disappearance of a JNDI name lookup when deploying a J2EE Web application!

Finally, wglance can be used to verify the successful first-time installation of WAS Advanced Edition. On the server where you have installed WAS AE, start the application server, then execute the following command:

```
# wglance wasaes <hostname-or-IP-where-you-
installed-WAS>
```

This command will verify whether or not your installation was successful by printing diagnostic commands to the terminal. For information on setting the wasmonhelper or wglance commands, refer to www.tcnd.com/p9.

Conclusion

In Part 2 of this series I will discuss how to write recovery scripts that can remotely compile and deploy Web applications on the tcnd.com network. Because such scripts can do the job without the intervention of an operator, I will show you how you can use them with WASMON to initiate the server takeover of a failing WebSphere node.

WebSphere DEVELOPER'S JOURNAL

Advertiser Index...

COMPANY	URL	PHONE	PG
CANDLE	WWW.CANDLE.COM/WWW1/HENRY1_WSDJ	800-898-4426	5
DIRIG SOFTWARE	WWW.DIRIG.COM/CORP/WP_REQUEST.HTML	603-889-2777	25
GLOBAL KNOWLEDGE	WWW.GLOBALKNOWLEDGE.COM	800-COURSES	34
IBM	WWW.IBM.COM/WEBSPHERE/OPENTOOLS	800-426-4968	11
KENETIKS	WWW.KENETIKS.COM	888-KENETIKS	17
MACROMEDIA	WWW.MACROMEDIA.COM/GO/WSDJ/	415-252-2000	21
MQSOFTWARE	WWW.MQSOFTWARE.COM	952-345-8720	6,7
PROLIFICS	WWW.PROLIFICS.COM	800-675-5419	37
QUEST SOFTWARE	HTTP://JAVA.QUEST.COM/JCSV/WDJ	800-663-4723	19
QUEST SOFTWARE	HTTP://JAVA.QUEST.COM/PERFORMANCE/WDJ	800-663-4723	51
REPORTINGENGINES	WWW.REPORTINGENGINES.COM/DOWNLOAD/WSDJ1.JSP	888-884-8665	13
SYS-CON PUBLICATIONS	WWW.SYS-CON.COM/2001/SUB.CFM	888-303-5282	49
TRILOG GROUP	WWW.FLOWBUILDER.COM	800-818-2199	52
VERSATA	WWW.VERSATA.COM/BUSINESSLOGICDESIGNER	800-984-7638	2
WILY TECHNOLOGY	WWW.WILYTECH.COM	888-GET-WILY	3

While there is no silver bullet, it can be done

Reducing the 80/20 Rule and Increasing Developer Productivity

BY ADAM KOLAWA

The 80/20 Rule is a well-known rule of thumb within the software development community that simply states that developers spend 80% of their time debugging applications and 20% writing new code. This ratio, which would seem to some outside the software industry the very embodiment of bad productivity, is unique to the software development community. No other industry measures work performed versus the amount of error fixing that needs to take place. Can you imagine what the production numbers would be for the Big Three automakers (GM, Ford, and DaimlerChrysler) if they spent 80% of their time fixing defects instead of making new cars? They would hardly be making any cars at all, and those they did would be utterly compromised in the eyes of the consumer.

ABOUT THE AUTHOR
Adam Kolawa is CEO, chairman, and a cofounder of Parasoft, a company that creates value-added products that improve the software development process. Adam is a well-known writer and speaker on industry issues and in 2001 was awarded the Los Angeles Ernst & Young Entrepreneur of the Year Award in the software category.

E-MAIL
ak@parasoft.com

There are those who find this rule so unnerving that they dismiss it with zealous disdain. The 80/20 rule applies only to "badly organized development groups" or "unskilled amateur coders," so the argument goes; it is something that happens to "other" development shops, but not something that can happen to "our group." The reason it is commonly dismissed out of hand is that it is felt that only the most unaware and incompetent software development team could miss such a staggering sign that their group is performing so inefficiently. Yet these dissenters are in a distinct minority; the 80/20 Rule is more or less accepted within the software development industry because programmers really do spend most of their time correcting bad code rather than producing new work.

Chances are pretty good that you think the 80/20 Rule doesn't apply to you and your development group. However, as incredible as this ratio is, I would bet that there are signs somewhere within your development organization that suggest you and your fellow developers really do spend a disproportionate amount of time debugging code rather than creating new work. Is your group showing symptoms of such poor productivity? Would you even recognize these signs if you saw them? It's harder than you think to recognize them.

A small experiment will confirm this hunch. Go and talk with your fellow developers while they are working and see if they consider their work "new" or something that they are trying "to make work." Walk in at different times during the day and see what this ratio is. Better yet, do this every day for several weeks across the entire group and see what the ratio of "working on new code" to "fixing code" is. Be sure to include your own work in this mix. I think you'll find that the 80/20 Rule does apply to your group. I also think you'll find, much to your own surprise, that your fellow developers consider the time they spend fixing code – "making it work" as they say – as actual coding time. They probably don't make a clear distinction between the two activities and they certainly won't be thinking about the 80/20 Rule as a strict demarcation of their time, if they think about it at all. If this is the case – if you and your comrades don't think of fixing and coding as two separate issues – then you have a big problem on your hands.

Once this experiment is conducted, you'll have enough data to take back to your fellow programmers and to your project managers to demonstrate how the group is really spending its time. Hopefully the entire group, managers and developers alike, will recognize and appreciate this data and will be eager to change the pattern of their development time. Realistically, it is the job of your project manager to help your group do something about the ratio of time they spend on new code versus debugging. However, there are a few things that you as a developer can do to help reduce the 80/20 ratio and increase your group's productivity.

The first step you should take to remedy the effects of the 80/20 Rule on your group is to think about how you are preventing errors. Review your work environment and the tools you are using. Are your tools sufficient for the kinds of applications you are writing? Do they help improve programming conditions and reduce errors? A good code editor is mandatory – the tools integrated into WebSphere Studio are a perfect example of a comprehensive IDE that helps developers write better code by providing a robust development environment for building and testing software applications before release. If you've got the right tools, then you are on the right track for reducing the 80/20 Rule.

However, if you don't feel that you have the proper tools, or if developers pick and chose their tools by personal preference rather than by group need, then perhaps this is a good indication that there is a communication breakdown between the development group and management. This is an easily forgotten point; not all errors are generated when code is written. If developers and their managers are not communicating – if you find it difficult to get the message across that you are not equipped properly for your tasks at hand – then chances are very good that the entire group is working under a flawed process and that steps should be taken to make intra-group communication easier and more proactive to developer needs. If your development group is fairly new, or you have recently been merged into another group or organization, a lot of your 80/20 problems could stem from simple process definition issues.

The next, and more important, step you should take is to look at your own needs, and those of your fellow developers, and compare them to the needs of the entire development group. How are the needs of individual developers different – or the same – from those of the group as a whole? Where do these needs intersect? To effectively address the 80/20 disconnect, you must begin to think in terms of group needs rather than in terms of individual developer needs. For the needs of the group, you must ask if the tool that is right for the individual is also right for the entire organization. Can the IDE you like to use for your own code be used to define group behavior? This is where a comprehensive plan for error prevention can begin to take root. As a group, are you taking full advantage of such tools, using them in the correct way, not just to find and fix errors, but also to prevent errors from recurring? Is the group using a comprehensive set of coding standards? Are you using these coding standards as a filter for source-control check-in? Many of the errors that propagate in the software development life cycle come to life through inconsistencies in the way developers work on their individual projects versus their group work dynamics. It won't matter if everyone uses the same tool if they all use it in a different way; i.e., if half of your fellow developers use a standard set of coding rules and the other half uses customized rules. Errors are bound to slip through if these rule sets are at all different.

WebSphere Studio is again a good example of an IDE that allows group behavior to be predefined, monitored, and refined in an effort to eradicate errors. Parasoft has an active partnership with IBM to provide WebSphere Studio with a variety of plug-in tools that extend the capabilities of WebSphere Studio for different types of application development, such as for Java applications or Web services or Web development projects. Such partnerships and plug-in tools reduce the probability of errors being generated across the development group through the use of language-specific coding standards and automated unit tests that can be shared among programmers and altered or revised as needed.

Reducing, or even reversing, the 80/20 Rule cannot, and will not, happen overnight. Looking at the dynamics within your development group and knowing that you and your fellow programmers have the right tools may be the work of a moment, but integrating common error prevention practices for that tool throughout the group – and getting positive results– is a much longer and more difficult task. There is no silver bullet for solving the 80/20 dilemma. Numerous studies have shown that in order to get full reversal of the 80/20 Rule could take as many as eight years. Clearly, it takes skill and patience to organize a development group into an efficient working team and to get results for your efforts.

What is important to understand is that getting the upper hand on the 80/20 Rule is largely a matter of analyzing and changing group behavior to address error prevention rather than using simple error detection and correction. You cannot, and will not, get a handle on errors if you act on an individual basis to correct your own code and leave your fellow programmers "on their honor" to correct their own work. Group behavior must be focused on uniformly meeting common coding standards, unit/load/functional tests, and monitoring practices. Only when a development group approaches error prevention as the norm, rather than performing simple error detection and bug fixes on an as-needed basis, will the 80/20 ratio begin to fall.

This is not simple hyperbole or wishful thinking – the rule can, in my opinion, be reversed, and permanently so. There is no reason why developers shouldn't spend 80% of their time developing new code and only 20% of their time debugging that same code. Actually, I am much more optimistic than that – I see no reason why debugging shouldn't be reduced even further, to less than 10% of a developer's total time, if the right tools and practices are integrated into a development group and those tools and practices are maintained for optimal performance and usage. Finding the right tool, such as WebSphere Studio, is the first step. Using that tool uniformly across a development team or set of teams is the next logical, and most important, determination that you and your fellow developers will need to make.

"If you...don't think of fixing and coding as two separate issues – then you have a big problem on your hands"

Solutions, Not Technology, Drive WebSphere Products

Part 2 of a conversation with the WebSphere marketing team

Customer involvement at all stages of product development, including early access for independent solution vendors, is crucial to IBM's strategy for managing the WebSphere Application Server development process, according to the WebSphere marketing team, headed up by Scott Hebner, director of marketing for WebSphere infrastructure software.

The WebSphere marketing team includes Joe Anthony, program director for WebSphere Extensions marketing; Derek Bildfell, program director for business development; Aimee Munsell, program director for WebSphere Application Server; Bernie Spang, program director for WebSphere Studio marketing; and Stefan Van Overtveldt, program director for WebSphere Technical Marketing.

WebSphere Developer's Journal editor-in-chief Jack Martin recently had the opportunity to talk with Hebner and his team in a wide-ranging and exclusive discussion. This second installment focuses on customer and partner input, early access to new technologies, rapid adoption, and the naming process for new products.

WSDJ: Since you're the group that decides what WebSphere is going to be next – because WebSphere is a lot of different things to a lot of different people and it's a product that is still obviously metamorphosing – how exactly do you decide what's going into it next? What makes you pick grid computing, for instance, as something that should be part of an application server as opposed to some type of translator that would work with all PDAs. How does that happen?

Van Overtveldt: One of the key things is that there is an incredible amount of technology that becomes available to us from our research department.

WSDJ: Yes, I've been to Watson – you have a tidal wave of technology coming out of your research department.

Van Overtveldt: We've been the patent leader in the industry for 10 years in a row now. We work with customers more and more in solving real customer problems and getting technology out of this, which we then use to drive our products or drive open standards. For example, we have technology out with customers right now that allows companies to use Web services in a more business-oriented fashion. It's the concept of service provisioning that none of our competitors can match and that we already have in production. Then, obviously, there are the more traditional ways of meeting customer requirements…

WSDJ: I've done a lot of work in Watson over the years and in any given cubicle is a scientist working on a product; in the next cubicle there's another scientist working on a completely different product; and three cubicles down there are five scientists working on another product – there are thousands upon thousands of them there. So how do you decide if a product is something a customer wants?

Munsell: We've always spent a lot of time with customers and what we've done in the last two–three years is drive a much more customer scenario–based process. So rather than just ending up with a list of cool individual features customers might want, we really try to understand much more in depth the types of projects the customers are trying to do every day and what their experience needs to be – everything from trying out the product, to purchasing it to installing it, deploying new apps, and then managing those applications.

In addition we are getting customers and partners involved earlier in working with development in a joint process to see what features we need to drive in to make that experience real. The biggest challenge of managing the application server development process is that since we have gotten so popular, the number of requirements has just escalated, both from other parts of the company, and from customers and partners. We probably get to maybe one tenth of the different things that we would like to do in each product coming out.

Bildfell: In our prioritization, we also consider what we have to build into the product, and what is available from our business partners. For example, emerging solutions in many areas of Web services provisioning and management are available from many of our partners, such as AmberPoint and Wily Technology. Where there are reliable partner solutions to fill out our customer requirements, it becomes less critical to build this capability into our products. This is also a critical differentiator in our approach to Eclipse; we've defined and delivered the fundamental technology, and we encourage our partners to build out specific solutions to their selected customer segments.

WSDJ: Say you have 100 features and only 10 of them are going to make it – what makes the 10 truly based on the majority of customers asking for them? Because you are saying that what's going into your products is solution driven, rather than technologically driven. You are not putting things in just because you can; you are putting things in because people want them.

Munsell: The top items are the ones most customers need but also the ones that will have the biggest impact on helping them solve their problems. We have to do some projection into how new technologies can solve a problem better in the future.

Spang: One of the other things we rely on is our early contact with customers through mechanisms like our alphaWorks site, where our research and development labs can post early technologies before they are committed in any products. The visitors at the alphaWorks site know that; they know they're getting access to the bowels of the research labs. From that we get feedback. Building on that we have teams such as the jStart Team (which stands for jumpstart), which started in the early days of Java when they were going out to jumpstart our customers' knowledge of Java. That team has evolved over the years to work with XML when it first came out, and Web services when it came out. Now grid computing is a part of it.

These are teams that go out and work with customers who are piloting things, testing things out to see how this new technology will solve their problems effectively. We are doing it in such a way that the customer knows that they are piloting something, and we are seeing how it goes. That gives us direct feedback from the real use and real value that the customers are seeing or, as we sometimes find, not.

Bildfell: This early access to our technologies is key to the rapid adoption we have seen in the independent solution vendor (ISV) community. ISVs need to start building their solutions before the latest technologies are stable enough for deployment in production solutions. These early experiences by the ISVs and early-adopter companies that work with JStart really help us to remain the leaders in emerging technologies – such as Web services and autonomic computing – which become the industry-leading solutions of tomorrow.

WSDJ: So, it's solution driven.

Spang: Yes.

WSDJ: That leads me to a question I have always been interested in. How do you decide what to name something? What is that process? Does Scott come in one day and say, "that's what this thing is." How do you end up with these names?

Munsell: As one example, we recently introduced WebSphere Application Server - Express. We went into that process and looked at the midmarket customers and customers who are really in the very early stages of e-business and looked at what they needed. We started to talk about the benefits that they are going to get from a product like this, and asked "what do we want the name to mean to them?" A lot of the feedback was that this is something that is going to have to be fast and easy, get them going quickly, sort of an on-ramp to getting on the expressway, so to speak. We came up with a short list of candidates, beat it up a lot, and "Express" was where we came out.

WSDJ: Do you do focus groups on naming? After all, the computer industry is the worst place for acronyms.

Munsell: We do focus groups because the two things we are trying to balance are, first trying to come up with something unique and accurate, but then making sure it is plain enough that people can readily understand what it is.

WSDJ: Exactly.

Munsell: I think a lot of the names we started out with and thought were really great descriptions turned out to take a paragraph to explain to the focus groups, who are a proxy for your prospective customers. Not a good thing.

Spang: One of the big accomplishments of my team over the past two years was that we had to decide on the brand transition, the naming transition from our VisualAge for Java product into what is now WebSphere Studio.

WSDJ: Which is an excellent name, by the way.

Spang: Thank you. But it came with a lot of problems. We already had a product line of WebSphere Studio Professional and WebSphere Studio Advanced, which at the time were on a different code base and were totally focused on Web development, the Web pages part. The Java development programming was VisualAge for Java.

We were faced with an issue where we had done focus groups on the right name for a development environment and Studio is one that resonated and was used by others in the industry, so it was high on the list. WebSphere Studio made sense, but we would have to transition from the state where our salespeople and customers thought of WebSphere Studio as just a Web development environment to thinking of it as a complete development environment.

One of the painful lessons that I have learned in the past two years is what you just mentioned about acronyms: we didn't get ahead of the "WSAD" use internally in time – "WSAD," which is used by all the development team, the people who built the developer domain Web sites, the developers who are writing documents and white papers about the product – and now the marketplace knows WebSphere Studio as WSAD, and we are now very much focused on working to change that – which is going to take a lot of effort. It would have been better to do it up front. WSAD is short for WebSphere Studio Application Developer, which is one configuration out of five base configurations of Studio, and it doesn't take into account the many toolkits that plug into those. It doesn't get the whole WebSphere Studio message across, and we are now in a bit of a recovery to get everybody to think in terms of WebSphere Studio. One of the other things I learned about naming is that you can't please all the people all of the time.

The process that we go through is to collect names and do a lot of brainstorming. We actually get the development team, the marketing team, and the sales team together in a room. We do brainstorming up on the wall, then we do focus group sessions. I've attended a few on the other side of the glass. One of the things that came consistently from developers about our product was, "it doesn't matter what you name it; if it's not good, I'm not going to use it, and if it's good, I don't care what you call it.

WSDJ: That's true, but the way I look at it – unless they can understand what it is in the first place, what you end up with is a very obscure product that a lot of hard-core coders may be using, but you miss maybe 95% of the market.

Spang: That's why we ended up using WebSphere Studio – simply because people know what a studio is – and then we named the configurations for the developer role: an application developer, a device developer, a site developer.

WSDJ: Like "station wagon" – everyone knows what you are talking about.

Munsell: In the technology industry, in particular, one of the things that you have to do is look at what the acronym for a potential name spells. We had a few that got ruled out just for that.

Spang: Application Server Suite was right out.

WSDJ: I think you've answered my question about customers being involved in the process. It is obvious that they had input into everything that you're doing.

Munsell: I have always been impressed by the fact that everyone at IBM spends a lot of time with customers and is pretty obsessive about trying to meet their needs. That is a given. But it is also true that everyone has their own language and interprets things differently. One of the things we started doing that has made a big difference is that we have the cross-functional team – marketing, sales, development, support – in the room at the same time with early testers of the product or concept testers.

It is amazing the in-depth discussions that you get with the customers when you have all those different views in the room at the same time, as opposed to people walking away and interpreting what's meant by "easy," for example. We all know customers want rapid time-to-market. So in one session a tester was asked, "What is rapid?" Some people in development – based on certain experiences with customers – expected the answer to be three months. I thought maybe three weeks, and we had a midmarket customer in the room who pretty much said three days to get up and running with a new project.

"Where there are reliable partner solutions to fill out our customer requirements, it becomes less critical to build this capability into our products"

-Derek Bildfell

PROLIFICS

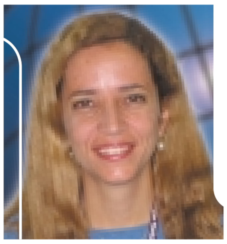
WWW.PROLIFICS.COM

The WebSphere Application Server 5 Web Services Technology Preview

Get a head start on the J2EE 1.4 APIs

– BY DENISE GABARDO AND ANDRÉ TOST –

Last month we described two new specifications that define the handling of Web services in J2EE, JSR101/ JAX-RPC, and JSR109. Both will be part of the J2EE 1.4 release, which is scheduled to go public by the end of the summer. In this article, we will show an example of an implementation of both new standards, which is provided in the Web Services Tech Preview for WebSphere Application Server 5.



ABOUT THE AUTHOR
Denise Gabardo has been working in application development since 1985 and has been a member of the IBM Software Group since 1996. Her main responsibility is to provide technical sales support to customers using WebSphere Application Server and WebSphere application development tools such as WebSphere Studio Application Developer and VisualAge for Java.

E-MAIL
deniseg@br.ibm.com

This tech preview comes as a free download from the Web, giving you a head start on the new APIs. We will show you two examples, one that takes you through the creation of a Web service that is offered to external clients, and a second showing how to develop a client to an existing Web service.

We will assume that you are familiar with the basic Web services technologies like SOAP and WSDL. See the Resources section at the end for pointers to more material on those topics. We will also assume that you have read our previous article, “Web Services Standards” (***WSDJ,*** Vol. 2, issue 3), describing the JSR101 and JSR109 specifications, or are otherwise familiar with them.

The WAS 5 Web Services Tech Preview

WebSphere Web Services for J2EE Technology Preview is based on the open-source Apache Axis project as a runtime engine, but also includes some enhancements, such as specialized serializers/deserializers for complex objects to obtain better performance. Axis is SAX-based, which makes it faster than implementations based on DOM parsers. Moreover, the deployment model is different; it follows the JSR109 specification, which Axis does not support.

It is important to note that the preview does not “support Axis”; instead, it supports the new standards and reuses some of the Axis work where it makes sense.

To obtain the tech preview installable package (approx. 10MB), download it from the WebSphere Developer Domain site at www7b.boulder.ibm.com/wsdd/downloads/web_services.html. Before installing it, you must already have either WebSphere Application Server (WAS) version 5 or WebSphere Application Client version 5 installed. The installation process is pretty straightforward; simply follow the instructions.

For our example, the generated EAR (enterprise application archive) file will be deployed using the usual WAS deployment mechanisms, either through the Admin Console, or via the wsadmin tool. The tech preview installation modifies the WAS administration code so that there are two additional “dialogs” in the WAS Admin Console (and additional prompts using wsadmin). We’ll show you what they look like later in the article.

There is not much more to say about the tech preview in particular. It implements both JAX-RPC and JSR109, so all the things we described in our earlier article are supported. Having said that, let’s dive into a concrete example.

Creating a New Web Service

There are different approaches to creating a new Web service, depending on your environment and your requirements. If some business function implementation already exists (in the form of a JavaBean or EJB), you would use the so-called bottom-up approach (see Figure 1). The tech preview provides a tool called “Java2WSDL” to generate the WSDL description from existing Java code.

On the other hand, a case may arise in which your Web services interface has already been defined in the form of a WSDL document (for example, if a service interface has been standardized for a particular industry). In this case,

you would use the so-called top-down approach (see Figure 2). Again, the tech preview offers a tool to handle automatic generation of code and other artifacts. This tool, called WSDL2Java, not only generates the appropriate deployment descriptors, it can also create the service endpoint interface and an implementation class with stubs for the required methods.

We will take you through an example using the bottom-up approach. We will assume that we have a class called WorkOrderManager that lets you create new WorkOrder objects or retrieve an existing WorkOrder by its number. We will then generate a WSDL file for this Java code using the Java2WSDL tool, followed by the WSDL2Java tool to create the deployment descriptor files. Finally, we will assemble all of these into a Web module and EAR, which we can install into the application server.

The first artifact that we will create is the service endpoint interface (SEI). It represents the Web service's port type to be published and contains all the methods we want to expose (see Listing 1).

Note how the interface extends java.rmi.Remote and how every method throws a java.rmi.Remote Exception. This is mandated by the JAX-RPC specification.

Next is the source code for the WorkOrder class (see Listing 2). What you should note here is that the class implements java.io.Serializable. This is a requirement for a nonbasic Java type that appears on the interface of a Web service (similar to parameters on an EJB's remote interface).

The WorkOrder class will be mapped to an XML Schema by the Java2WSDL tool. This schema then becomes part of the WSDL definition for our Web service.

Finally, Listing 3 shows the code for the WorkOrderManagerImpl class, which implements the service. We kept the code very simple, because we want to focus on the creation of the Web service artifacts here. A real business application would certainly look different.

After you have compiled these three classes (WorkOrder, WorkOrderManager, and WorkOrderManagerImpl), you are ready to generate the WSDL document.

The Java2WSDL Tool

The first of the tech preview tools that we will use is the Java2WSDL tool, which generates the WSDL file from the SEI class. It is located in the <was_install_root>\bin directory.

We assume here that the code exists in the c:\was5tp\tech\preview directory, and that you have added c:\was5tp to your classpath. Make c:\was5tp your current directory and enter this command:

```
java2wsdl tech.preview.WorkOrderManager
```

This will create a WSDL file called WorkOrderManager.wsdl. You will receive a message indicating that "the –location was not set,...". The location point of the Web service will be specified when you install the application in WAS, so you can ignore this message for now.

We have now created the contract that our Web service exposes and can use as the input for creating all other artifacts needed to deploy the service.

The WSDL2Java Tool

We will now use the WSDL2Java tool with the generated WorkOrderManager.wsdl file as input to generate the Web service deployment descriptor templates:

```
wsdl2java –verbose –META-INF-Only –server-
side Bean –output . WorkOrderManager.wsdl
```

This will create two directories, META-INF and WEB-INF, with the following Web service deployment descriptor files:

- ***META-INF:***
  - –*webservicesclient.xml:* This file is the deployment descriptor for any client that uses the Web service and runs in a J2EE client container.
  - –*WorkOrderManager_mapping.xml:* This file contains the mapping between a namespace that is used in the WSDL document and the Java package that is used in the Java code.
  - –*ibm-webservicesclient-bnd.xml:* This file is used to create definitions for securely accessing the Web service.
- ***WEB-INF:***
  - –*webservices.xml:* This is the Web service deployment descriptor that defines the new Web service to the application server.
  - –*WorkOrderManager_mapping.xml:* This file is an exact copy of the one created in the META-INF directory.
  - –*ibm-webservices-bnd.xml:* Again, this file is used if you want to secure your Web service.

Generally, the files in the META-INF directory are used for a client to the Web service, whereas the files in the WEB-INF directory are used for server-side deployment. The only files defined in the standard are the webservices.xml and webservicesclient.xml deployment descriptors; the other files are specific to WebSphere.

The only modification you need to make is in the webservices.xml file. Make this change :

```
<servlet-link>WorkOrderManager</servlet-link>
```

This binds the Web service to a particular implementation, the WorkOrderManager class. We will reuse this

ABOUT THE AUTHOR

André Tost works as a solution architect in the WebSphere Business Development group, where he helps IBM's Strategic Alliance Partners integrate their applications with WebSphere. His special focus is on Web services technology throughout the WebSphere product family. Before his current assignment, he spent 10 years in various development and architecture roles in IBM software development, most recently for WebSphere Business Components. Originally from Germany, he now lives and works in Rochester, MN.

E-MAIL

atost@us.ibm.com

value later when we assemble the application (i.e., it will go into the web.xml deployment descriptor for the Web application).

Creating the EAR File

Now we can go ahead and assemble the generated files into an archive file that we can install into the application server. Do this using the Application Assembly Tool (AAT) that comes with WAS. You can start it by entering "assembly" on the command line.

Choose to create a new Web module. Since this is all standard AAT business, we won't describe it in detail here and will simply refer to the AAT documentation for more info. Name the new Web module "WorkOrderManager Web.war".

Add all of the class files to the new Web module. They go into the \WEB-INF\classes\tech\preview directory. Moreover, you need to add a new Web component, a servlet. Its name is WorkOrderManager (this is referenced from within the webservices.xml file) and its class is tech.preview.WorkOrderManagerImpl. Note that even though this class is not really a servlet, it is defined as one in the Web module deployment descriptor. During deployment the application server will make this class accessible as a Web service (by effectively wrapping it into a servlet that receives the SOAP request and invokes the appropriate method on the bean). Save the new Web module and exit the AAT.

Now move the WorkOrderManager.wsdl file into the c:\was5tp\WEB-INF directory. On the command line, type the following command to add the remaining files to the module:

```
jar –uvf WorkOrderManagerWeb.war WEB-INF/*
```

Using the AAT again, create the EAR application (WorkOrderManager.ear) and import the Web module WorkOrderManagerWeb.war. Make its context root "/workorder". This is all business as usual and does not require any special steps for the Web service.

Installing the Application

Now you are ready to deploy and install the enterprise application into WebSphere Application Server. As we already mentioned, the installation process is similar to that of any other enterprise application. After installing the technical preview, there are two additional steps. First, the system asks for the location to place the WSDL files. Enter "c:\was5tp" into that field. This will export the resulting WSDL file to the c:\was5tp\WorkOrderManager.ear\WorkOrderManagerWeb.war\WorkOrderManagerService directory. We will need it later when we create the client.

The second new step requires information about the protocol, hostname, and port on which the Web service will run. You can leave all the defaults. The WSDL file in the Web module is updated with the new endpoint, and a copy is exported to the defined directory.

After installing the enterprise application, save the configuration, start the WorkOrderManager.ear application, and your Web service is ready to go!

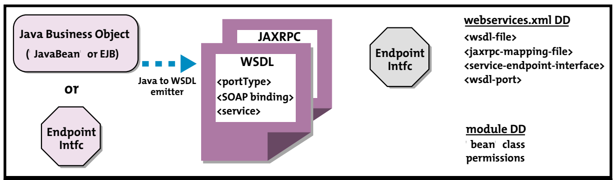


FIG. 1: DEVELOPING A WEB SERVICE: BOTTOM-UP

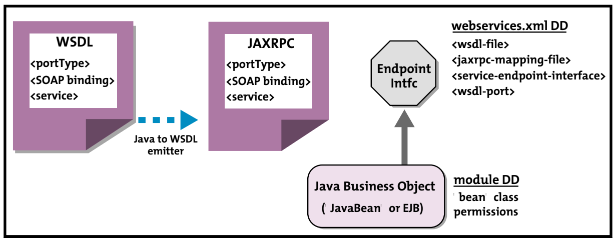


FIG. 2: DEVELOPING A WEB SERVICE: TOP-DOWN

Accessing an Existing Web Service

Now that you have a Web service ready and waiting to be called, we want to create a client to use this service. We will access the service that we just created; however, this could also be any other Web service described by a WSDL document.

The client interface we will use is based on the JAX-RPC standard, as described earlier. A JAX-RPC–compliant Web service client can run stand-alone or in a J2EE client container. We will keep things simple here by running the client from the command line.

The only input we will need is the exported WSDL file from the installation of the enterprise archive with the Web service. Feel free to delete the files generated before; they are all safely installed in the application server and we don't need them here anymore. Copy the WorkOrderManager.wsdl file to the c:\was5tp directory. Now use the WSDL2Java tool again to create the client-side bindings for the service: wsdl2java WorkOrderManager.wsdl

This will create the classes we need on the client side. All of the generated Java files go into the tech\preview directory:

- ***WorkOrder.java:*** This class is generated from the XML Schema for WorkOrder in the XML Schema.
- ***WorkOrder_Deser.java., WorkOrder_Ser.java, WorkOrder_Helper.java:*** These classes serve as helpers to convert an XML document into a WorkOrder object and vice versa.
- ***WorkOrderManager.java:*** This is the interface of the Web service. (Yes, we had these classes before, but remember that we generate all of this only from the WSDL file.)
- ***WorkOrderManagerService.java:*** This interface extends the javax.xml.rpc.Service interface and provides some convenient methods to obtain proxies to the Web service dynamically.
- ***WorkOrderManagerServiceLocator.java:*** This class is an implementation of the foregoing interface.
- ***WorkOrderManagerSoapBindingStub.java:*** This class represents the actual proxy, but you will never use it directly in your client program.

The WSDL2Java tool created a number of deployment descriptor files in the META-INF directory. These files would be needed if we were to access the Web service from within a J2EE client container. We already described them earlier when creating the server-side deployment descriptions. We won't need them here, because we will run the client from the command line. (In Java standards lingo, we are using a "J2SE client.")

Go ahead and compile all of the generated Java files. To do so, you need the following set of JAR files in your classpath. They can be found in the <was_install_root>\lib directory: axis.jar, qname.jar, jaxrpc.jar, j2ee.jar, commons-logging-api.jar, commons-discovery.jar, and saaj.jar.

And finally, Listing 4 is a small test application that uses the Web service, which creates two new WorkOrder objects and then retrieves one of them.

Conclusion

The new standards for providing and consuming Web services will help developers create portable and interoperable Web services applications across a variety of application servers. While this is coming later this year in the J2EE 1.4 specification, IBM provides you with an early implementation of the new standards on top of WebSphere Application Server 5, allowing you to get a head start on becoming familiar with the new APIs. Also later this year, complete tooling will follow as part of WebSphere Studio Application Developer.

Resources

- *JAX-RPC:* http://java.sun.com/xml/jaxrpc
- *JSR109:* http://jcp.org/en/jsr/detail?id=109
- *The WAS 5 Web Services Tech Preview:* www7b.boulder.ibm.com/wsdd/downloads/web_services.html
- *WebSphere Version 5 Web Services Handbook:* http://publibb.boulder.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg246891.html?Open
- High, R.; Herness, E.; Rochat, K.; Francis, T.; Vignola, C.; and Knutson, J. (2003). *Professional IBM WebSphere 5.0 Application Server.* Wrox.
- *The IBM developerWorks Web services zone:* www.ibm.com/developerworks/webservices

LISTING 1: THE SERVICE ENDPOINT INTERFACE

```
package tech.preview;

public interface WorkOrderManager extends java.rmi.Remote {

  public Integer createNewWorkOrder(String
     customerName,String sourceCompany)
         throws java.rmi.RemoteException;

  public WorkOrder getWorkOrder(Integer orderNumber)
         throws java.rmi.RemoteException;
}
```

LISTING 2: THE WORKORDER CLASS

```
package tech.preview;

import java.io.Serializable;

public class WorkOrder implements Serializable {
  private String customerName;
  private String sourceCompany;
  private Integer orderNumber;
  // manage a sequence to assign order number
  static Integer nextOrderNumber = new Integer(0);

  public WorkOrder() {
         this.orderNumber = nextOrderNumber;
         nextOrderNumber=new
           Integer(orderNumber.intValue()+1);
  }
  public String getCustomerName() {
         return this.customerName;
  }
  public void setCustomerName(String customerName) {
         this.customerName = customerName;
  }
  public String getSourceCompany() {
         return sourceCompany;
  }
  public void setSourceCompany(String sourceCompany) {
         this.sourceCompany = sourceCompany;
  }
  // no set method - the order number is created by the
  // constructor and cannot be changed later
  public Integer getOrderNumber() {
         return this.orderNumber;
  }
}
```

LISTING 3: THE WORKORDERMANAGERIMPL CLASS

```
package tech.preview;

import java.util.Hashtable;
public class WorkOrderManagerImpl implements WorkOrderManager
{

  static Hashtable orders = new Hashtable();

  // Get input data and create a new work order.
  // Return the order number

  public Integer createNewWorkOrder(
         String customerName,
         String sourceCompany) {

         WorkOrder newOne = new WorkOrder();
         newOne.setCustomerName(customerName);
         newOne.setSourceCompany(sourceCompany);
         orders.put(newOne.getOrderNumber(), newOne);

         return newOne.getOrderNumber();
  }

  public WorkOrder getWorkOrder(Integer orderNumber) {


         return orders.get(orderNumber);
  }
}
```

LISTING 4: TEST APPLICATION

```
package tech.preview;

public class TestWorkOrder {

  public static void main(String args[])
  {
         try
         {
                  WorkOrderManager mgr =
                          new WorkOrderManagerService
                          Locator().getWorkOrderManager();
                  int i1 = mgr.createNewWorkOrder("Jim
                    Smith", "ACME");
                  System.out.println("Created Order with
                    number "+i1);

                  int i2 = mgr.createNewWorkOrder("Bill
                    Myers", "ACME");
                  System.out.println("Created Order with
                    number "+i2);

                  WorkOrder order = mgr.getWorkOrder(i1);
                  System.out.println("Retrieved order with
                    number "
+order.getOrderNumber()
                                                 +" and source company "
+order.getSourceCompany());
         }
         catch (Exception x)
         {
                  System.err.println("Exception : "+x);
         }
  }
}
```

Hynes Convention Center: Boston, MA

Web Services Edge 2003 East

INTERNATIONAL WEB SERVICES CONFERENCE & EXPO

Russ' Tool Shed: Russ Fustino shows how to use Visual Studio .NET

When SYS-CON Media's sister company, SYS-CON Events, began preparing last year for this spring's "Web Services Edge" Conference & Expo, one consideration was paramount: every effort in the nine-month preparation cycle should be geared toward making it indisputably the world's largest independent Java, .NET, XML, and Web services event.

That particular mission was accomplished on March 18–20, 2003, at the centrally located Hynes Convention Center in Boston, Massachusetts, when Web Services Edge 2003 East made its mark right from the get-go, with delegates from a wide variety of companies both technologically and geographically. Not only had they been attracted by the specific session tracks for Java, .NET, XML, and Web services, they had also come to take advantage of the all-day *i*-technology tutorials, whether it was the Sun Microsystems Java University, the IBM XML Certified Developer Fast Path, Russ Fustino's .NET workshop (Russ' Tool Shed), or Derek Ferguson's Mobile .NET tutorial.

The show opened with a very well-attended keynote from Oracle's John Magee, VP of Oracle9*i* Application Server. Magee stressed that the key to understanding why Web services, unlike its distributed-computing forerunners like COM and CORBA, is prevailing in the enterprise space is that Web services do more than merely enable interoperability between platforms and integration between applications – they also do so simply.

What drives their simplicity, Magee explained to the audience, is standards.

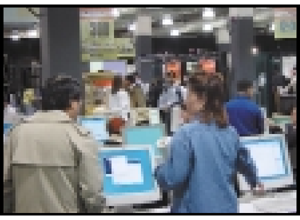
The afternoon keynote offerings on Day One of the conference were equally well received. First came a panel coordinated by the Web Services Interoperability Organization (WS-I). The WS-I is an open industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages, and the panel discussion took place against the backdrop of the WS-I Basic Profile 1.0, consisting of a set of non-proprietary Web services specifications. The working draft for this, the audience learned, was approved just four weeks before the conference.

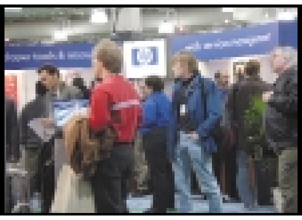
But security, the panel agreed, was the primary priority. Now that corporations like Merrill Lynch and DaimlerChrysler have joined the organization, ensuring that everyone adheres to the same specification is more important than ever. Web services is moving beyond mere SOAP, WSDL, and UDDI toward addressing security, messaging, reliability, and transactions. Eric Newcomer, chief technology officer of IONA Technologies, emphasized the importance of the World Wide Web Consortium (W3C) approach to these challenges, an effort that centers on the W3C's Web Services Specification Effort.

The Web services keynote panel was quickly followed by the highlight of Day One for many of the delegates gathered in the keynote hall: an address by Miguel de Icaza, the impossibly young and extremely gifted founder and leader of the GNOME Foundation, cofounder of Ximian, Inc., and .NET expert extraordinaire – as anyone needs to be who leads a project designed to port .NET to the Linux operating system.

The Mono Project, as de Icaza's project is called, clearly fascinated the broad mix of developers attending the conference.

After explaining that GNOME – a desktop development platform and suite of productivity applications – is his compa-





ny's key focus and is mostly developed in C, C++, Python, and Perl, he went on to recount how for every new GNOME API (GNOME is component-oriented and supports many programming languages), GNOME developers needed to develop language-specific bindings. Thus .NET, which also addresses the multilanguage problem, was of immediate interest to de Icaza.

As soon as he learned about the .NET Framework, he told the spellbound audience, he got excited – a single Virtual Execution System for multiple languages, with a large and reusable factored class library, that was, in his view, just what was needed. As well as being a new way to do things, .NET's rich support for interop (COM, P/Invoke) meant you didn't have to rewrite everything all at once.

And so Mono was born: an open-source .NET Framework implementation.

It's based around the CLI ISO standard, de Icaza continued. It has a CLI-compliant execution system and a x86 JIT compiler. It's supported by Windows, BSD, Linux, and Solaris, and there has been lots of progress on the class libraries.

The Windows support, de Icaza said, was merely a function of the fact that 60% or so of Mono developers have a Windows background. Some of the code contributed to Mono was funded by Microsoft grants, he added.

At the end of his keynote address, scores of developers of every stripe got up from their chairs and surrounded de Icaza for further questions. The response to his good humor, rapid delivery, technical savvy, and sheer charm had been overwhelming and with his keynote, Web Services Edge 2003 (East) passed a significant milestone: no previous conference in the series had ever included so wide a range of technical content.

Day Two saw Sun's Mark Herring take the keynote stage and his mastery of the whole Web services paradigm was clearly in evidence. Extended coverage of both his Java keynote and the subsequent keynote address by Jesse Liberty are available on the main conference Web site, www.sys-con.com/WebServicesEdge2003East.

The closing keynote discussion panel, which for many turned out to be the high point of the entire keynote program, was a wide-ranging and a sometimes heated debate about "The Future of Java." The whole intense and highly interactive hour exemplified very well how a SYS-CON *i*-technology conference program differs from that offered by any other conference organizer. This was panel discussion at its best.

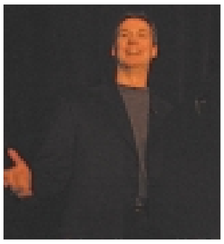
True to the enormously close links that ***Java Developer's Journal*** enjoys with the software development industry, the participants in this final panel at Web Services Edge 2003 (East) had come to Boston from far and wide. Sun's chief technology evangelist Simon Phipps had flown over from the UK and BEA's director of technology evangelism Tyler Jewell had traveled from Los Angeles. Sonic Software's VP and chief technology evangelist Dave Chappell may have nipped across from Bedford, MA, but Aligo's CTO Jeff Capone had flown in from San Francisco, and JBoss founder Marc Fleury had come up from the JBoss Group's company's HQ in Atlanta, Georgia.

We fully expect the next Conference & Expo, Web Services Edge (West) in October, to be equally chock-full of the movers and shakers of the software development industry as it continues its headlong progress toward distributed computing with full application integration and interoperability.

All in all it was a marvelous conference, and the Expo hall was intensely busy from the moment it opened to the moment it closed two days later.

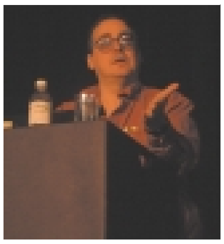
This is not the end of the Web services "story," nor is it even the beginning of the end; but March 18–20 in Boston's Hynes Convention Center may well have marked the end of the beginning.

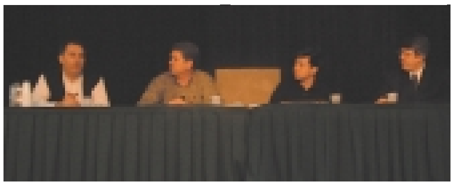
Come join us for Phase Two…in October in California.


John Magee, Oracle: "Developing in a Services-Based World"

Mark Herring, Sun Microsystems: "Bridging the Gap Between WS-Myth and WS-Reality"

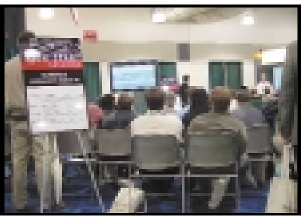
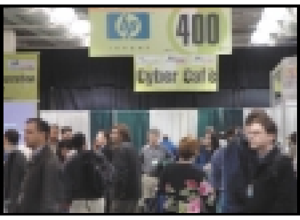
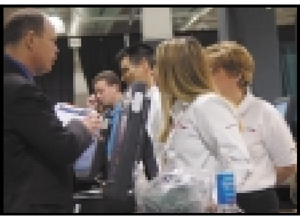
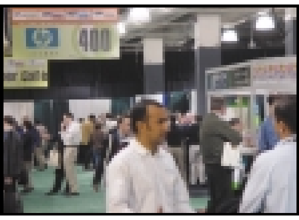
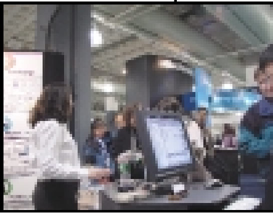
Miguel de Icaza, Ximian: "The Mono Project"


Jesse Liberty, Liberty Associates: ".NET Web Services"


WS-I Panel Discussion: "A Road Map for Web Services Standards"

.NET Panel Discussion: "Real-World .NET"

Java Panel Discussion: "The Future of Java"

CASE STUDY

Part 3: Applying patterns

Discovering and Documenting Business Application Patterns

BY BRENT CARLSON AND JAMES CAREY

Over the past two months, we've looked at the process of extracting a business application pattern from a series of business requirements. You've seen this pattern take shape, from its original form as a design meeting the specific needs of a particular business application (configurable product balance information) through an initial abstraction that was modified by other business requirements to reach its final form. In this article, the last of a three-part series, we'll look at applying the key pattern and cached balances pattern (with others) to the construction of applications, components, and Web services.

ABOUT THE AUTHOR

Brent Carlson is vice president of technology and cofounder of LogicLibrary, Inc., a leading provider of enterprise software and services for enabling existing software development assets as Web services. With James Carey, Brent has coauthored *SanFrancisco Design Patterns: Blueprints for Business Software* and *Framework Process Patterns: Lessons Learned Developing Application Frameworks* (Addison-Wesley).

E-MAIL

brent.carlson@logiclibrary.com

Reviewing Our Pattern

Last month's article concluded with a definition of our business application patterns: keys and cached balances. We tracked the maturation of these patterns through their initial concept as extracted from our product balances business requirement, to their modification due to additional requirements resulting from our application's need to flexibly manage financial account balances, to their representation both as abstract patterns and reusable partial implementations of each pattern (i.e., framelets). Our final patterns included an abstracted CachedBalanceSet class containing a set of balance values associated with abstracted access keys (the AccessKey class). These keys are controlled by another set of keys that specify their scope (the SpecificationKey class), and each of these key classes in turn contains a collection of key elements that implements an abstract Keyable interface that declares comparison methods such as equals (for AccessKeyable) and convert (for SpecificationKeyable). Figures 1 and 2 reflect these concepts in terms of UML.

Patterns Within the J2EE Architectural Context

Up to this point, we have not taken into account the specific demands and restrictions placed upon our pattern by the technical architecture within which it will be deployed. In other words, our pattern provides a set of high-level design guidelines that need to be adapted for specific use. Our choice of technical architecture is J2EE, and more specifically EJB. How does the EJB framework affect our low-level design choices when applying this pattern?

As we enter into the detailed design phase, let's review the characteristics of our CachedProductBalances class (and related classes and algorithms):

- It represents a composite set of business information related to a specific business entity; namely the Product class.
- Its methods are self-contained; i.e., their implementations do not depend upon making calls to other business entities.
- Its methods will typically be called within the context of a broader business process (e.g., invoking updateBalance as part of incoming shipment processing) rather than as stand-alone business functionality.

These characteristics need to be evaluated in the context of the EJB framework and the design options that framework presents. We should also rely on the experience of others in building robust J2EE applications – after all, that's what design patterns are all about. Let's take a look at some key design aspects and see what *Core J2EE Patterns* has to say about them.

ENTITY BEAN DESIGN – THE COMPOSITE ENTITY PATTERN

One of the early decisions we need to make when applying our pattern to EJB technology is which classes should be implemented as entity beans. The Composite Entity pattern states "entity beans are not intended to represent every persistent object in the object model. Entity beans are better suited for coarse-grained persistent business objects." Applying this principle to our high-level design for CachedBalanceSet, we see that most classes within this design are best left as simple Java classes; in other words, they are dependent objects. Should the CachedBalanceSet class itself be implemented as an entity bean? Given the previously listed characteristics (specifically that the CachedBalanceSet method implementations are self-contained), we can make a reasonable case for implementing CachedBalanceSet as an entity bean. However, we also need to consider what its relationship to the Product class should be and how its methods interact with broader business processes, as these points may very well cause us to revisit our initial decision.

The Product class is a natural candidate for entity bean implementation. It represents a business entity with considerable information and functional behavior, and it presents us with a natural primary key – the product ID. If we implement the Product class as an entity bean, what effect does that have on our CachedBalanceSet class? Much depends upon the behavior we choose to expose on Product's interface. If we choose to treat the Product class primarily as a data holder, maintaining information such as product name, description, ID, and the like, without incorporating additional business process–related behavior (like product balance updating and retrieval), then we maintain a loose coupling between the Product class and the CachedBalanceSet class (see Figure 3). This lends credence to our tentative decision to implement the CachedBalanceSet class as an entity bean in its own right, but raises the question as to where to contain the business process–related behavior that manages product balances.

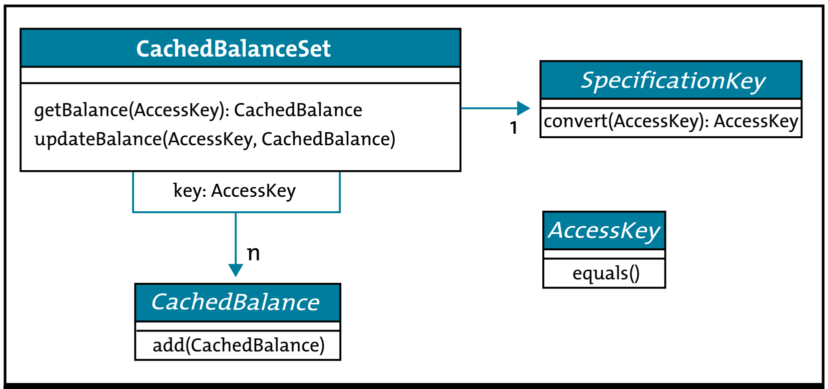
Another aspect of the EJB architecture that we need to take into account is our ability to choose whether an entity bean is defined as local. Local entity beans reduce overhead associated with remote calls but at the expense of limiting access to the entity object to the local process. At this point in our design process we don't have clear guidance based on the Composite Entity pattern, but as we continue to expand our design toward components and Web services, we will see that local entity beans are the best choice for our design.

If, on the other hand, we choose to embed product balance maintenance within the Product class, we see that we have introduced tight coupling between the Product class and the CachedBalanceSet class. Such tight coupling might cause us to reconsider our earlier decision to make the CachedBalanceSet class an entity bean, instead choosing to implement the CachedBalanceSet class as a dependent Java class within the Product class implementation (see Figure 4). However, this raises another question – how much overhead does introducing these dependent classes introduce to EJB activation and passivation? If our typical use of the Product class does not involve balance management, we might be better off separating the CachedBalanceSet class from the Product class as we originally proposed. If, on the other hand, most uses of the Product class involve balance management, then embedding the product balance management code as dependent objects within the Product class is probably the right decision. If we choose this option, we may be able to mitigate the EJB activation and passivation overhead through a bean-managed persistence (BMP) approach, deferring activation of the dependent CachedBalanceSet object within the Product EJB until it is needed. In fact, this is likely to be the choice we will make for CachedBalanceSets even if we choose to implement it as a separate entity bean, given that the internal intricacy of its contained objects, in this case as it is in general, involves a series of tradeoffs.

SESSION BEAN DESIGN – THE SESSION FACADE PATTERN

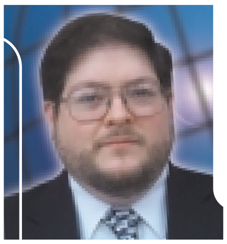
Let's assume that we have chosen the decoupled approach as described above, resulting in two entity beans: Product and CachedBalanceSet. Where does the business logic responsible for maintaining product balances then reside?

Again, *Core J2EE Patterns* gives us a strong hint. Reading from the Session Facade pattern, we see that a session bean "…manages the business objects and provides a uniform coarse-grained service access layer to clients." There are two important points being made here, one of which is directly applicable to our specific problem – session beans should be designed to manage underlying business objects. (We'll consider the second point when we discuss the relationship of Web services to components.) By introducing a WarehouseManagement session bean following the Session Facade pattern, we now have a natural place to locate our product balance maintenance implementation (along with many other business process–related algorithms that span products, warehouses, and other related business concepts such as lead and shipping time calculations (see Figure 5). By doing so, we have in fact created a coarse-grained component – one that presents a series of business services to



CachedBalanceSet
getBalance(AccessKey): CachedBalance
updateBalance(AccessKey, CachedBalance)
key: AccessKey
n
CachedBalance
add(CachedBalance)
1
SpecificationKey
convert(AccessKey): AccessKey
AccessKey
equals()

FIG. 1: CACHEDBALANCESET UML DIAGRAM



ABOUT THE AUTHOR

James Carey, senior software engineer, is lead architect for IBM's WebSphere Business Components project. This project develops EJB components that provide business content to support application development. With Brent Carlson, James has coauthored *SanFrancisco Design Patterns: Blueprints for Business Software* and *Framework Process Patterns: Lessons Learned Developing Application Frameworks* (Addison-Wesley).

E-MAIL
jecarey@us.ibm.com

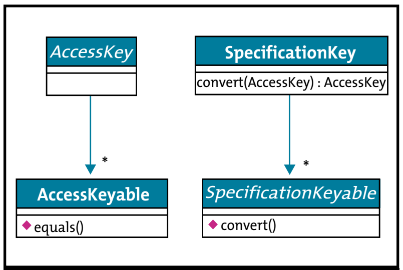


FIG. 2: KEY AND KEYABLES UML DIAGRAM

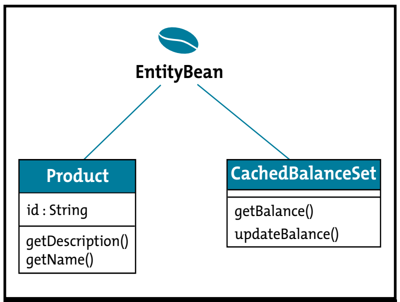


FIG. 3: CACHEDBALANCESET AND PRODUCT AS ENTITY BEANS

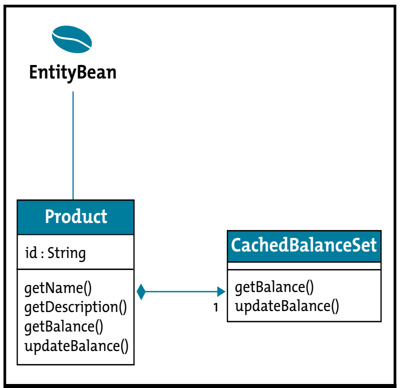


FIG. 4: CACHEDBALANCESET AS A DEPENDENT OBJECT WITHIN PRODUCT ENTITY BEAN

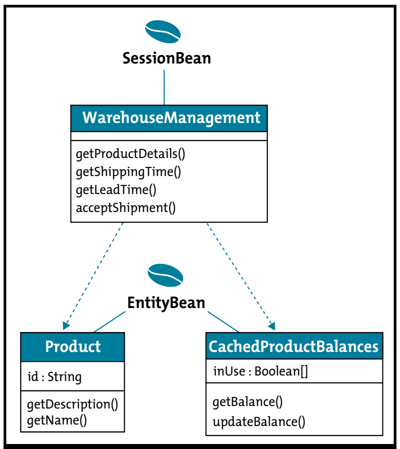


FIG. 5: SESSION FACADE AND RELATED ENTITY BEANS

clients without exposing those clients to the underlying complexity inherent in the implementation of those services. This approach also reinforces our earlier decision to separate product balance maintenance logic from the Product entity bean and maintain loose coupling between the Product and CachedBalanceSet classes.

Now that we've introduced a session bean into our design, we need to consider whether this bean should be stateful or stateless. In general, *Core J2EE Patterns* recommends stateless session beans, as this allows the application server to more efficiently manage its memory pool by allocating beans out of a bean cache on a method-by-method basis. If, however, there is a need to maintain client session state outside of the client itself, then a stateful session bean approach may be warranted. In our case, we will assume that the stateless session bean approach is appropriate.

Other design patterns may come into play as we drop down into the detailed design of our Warehouse-Management session bean. For example, we might choose to implement each of the methods our session bean exposes using the Strategy pattern, thus making our session bean highly configurable. Individual strategy implementations might in turn use the TemplateMethod pattern to provide a partially built algorithm that can be customized on a point-by-point basis. These are just a couple of examples of how design patterns can influence detailed design decisions.

Bringing Web Services into the Picture

SERVER-SIDE CONSIDERATIONS – THE TRANSFER OBJECT PATTERN

Now that we have our coarse-grained business services defined, we need to adapt them to the needs of potential clients. One type of client that is of considerable interest these days is remote invocation via a Web service. Web service–based invocation involves transmission of a client request to a remote service via an XML message whose format is specified by SOAP, typically transmitted over Internet protocols such as TCP/IP and HTTP. Introduction of this XML-based invocation into the picture adds some additional design restrictions above and beyond those created by the base J2EE architecture.

Specifically, because our client communicates with our service via XML documents rather than remote objects (as would be the case in an *n*-tier J2EE implementation with JSPs and servlets invoking session bean methods through their remote interfaces via RMI), we need to ensure that the interface presented by our server component is easily consumable by the SOAP client and does not introduce a lot of overhead at the component (i.e., session bean) interface.

The Transfer Object pattern provides us with an approach for consolidating the business information to be communicated between client and server. This pattern encourages the use of simple Java classes to encapsulate business data exposed on the public interface of a coarse-grained component. These simple data-oriented Java classes can then be easily processed by the serializer/deserializer logic provided by SOAP development and runtime frameworks such as those following the JAX-RPC specification.

CLIENT-SIDE CONSIDERATIONS – THE PROXY AND BUSINESS DELEGATE PATTERNS

Remote Web services clients typically interact with a service via object-oriented proxies that are often generated by specialized Web services development tools from the WSDL document describing the service. These simple proxy classes provide individual methods for each of the operations described by the WSDL document. Each method directly invokes the underlying service (in our case the coarse-grained component implemented by our WarehouseManagement session bean) via the SOAP-based framework provided by the Web services runtime.

In some cases, these directly generated client methods will be sufficient. There may be cases, however, in which it is desirable for the client to have an additional level of decoupling from the services it invokes. For example, we might choose to build up a cache of recently accessed product information to minimize remote activity and thus improve performance of our client, or we might want to build up a consolidated client view over multiple Web services. The Business Delegate pattern describes an approach that supports this level of isolation. Client activities occur solely through a client-side class, the Business Delegate, which in turn delegates any necessary remote invocations to the underlying business service (in this case our generated SOAP client). Because the Business Delegate class is interposed between the client code and the SOAP client, we have the freedom to introduce client-side caching for frequently accessed information, consolidated service groupings with embedded glue logic, or other useful features. The Business Delegate class can also serve as a stable interface point for the remainder of the client code, isolating any client-side changes that might result from changing server-side implementations to a single touch point.

Summary

Figure 6 shows our full set of patterns working in concert from client to server through a client request scenario. As you can see, by defining our pattern independently of the technical architecture, we were able to easily take advantage of a number of useful concepts:

- Grouping related business function into loosely coupled entities with embedded dependent objects – the Composite Entity pattern
- Providing cross-entity business process functionality via an encapsulating facade – the Session Facade pattern
- Consolidating the business information transferred between client and server into simple data-oriented objects – the Transfer Object pattern
- Accessing server-side functionality via client-side proxies – the Proxy pattern
- Introducing additional client-side isolation classes to support service consolidation, client-side caching, and other advanced client-side capabilities – the Business Delegate pattern

Underneath it all reside our original business patterns, doing the heavy lifting of managing our product balances.

Resources

- Alur, D.; Crupi, J.; and Malks, D. (2001). *Core J2EE Patterns: Best Practices and Design Strategies.* Prentice Hall.
- Gamma, E.; Helm, R.; Johnson, R.; and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.
- Carey, J. and Carlson, B. (2002). *Framework Process Patterns: Lessons Learned Developing Application Frameworks.* Addison-Wesley.
- Carey, J.; Carlson, B.; Graser, T.; and Varlson, B. (2000). *SanFrancisco Design Patterns: Blueprints for Business Software.* Addison-Wesley.

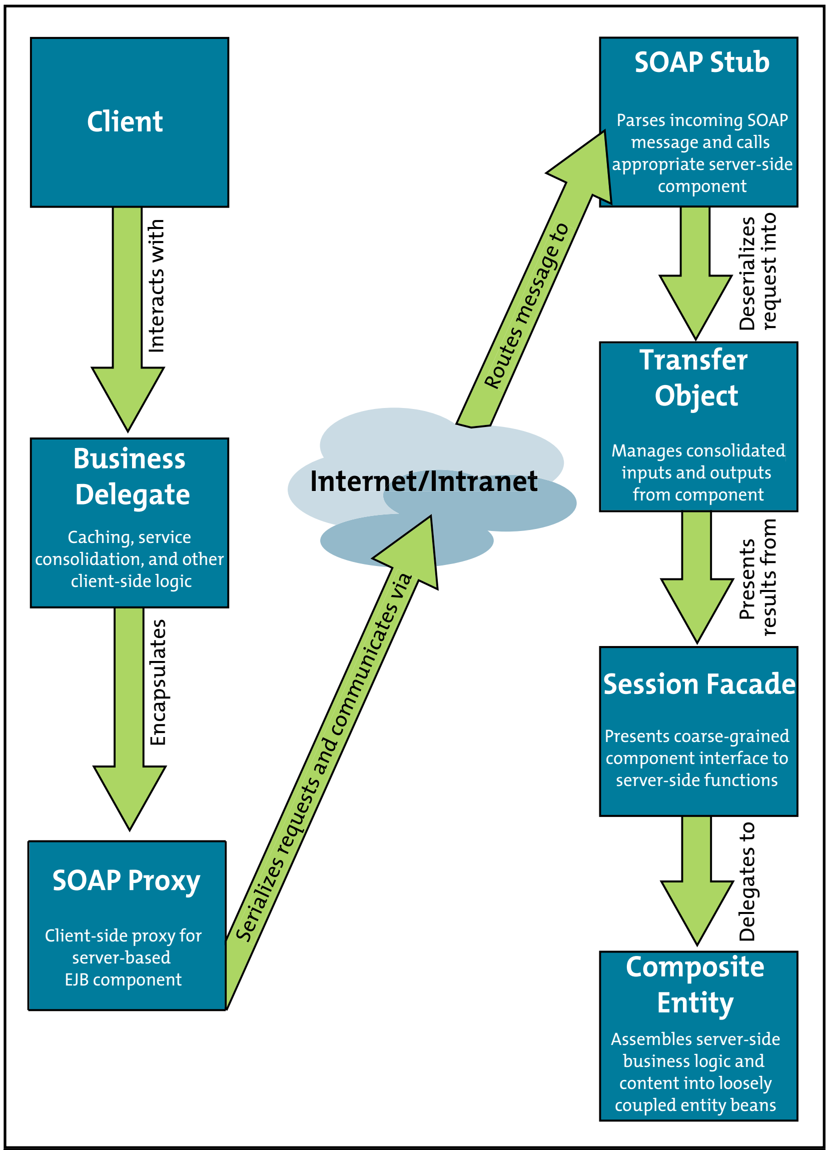

Client
Interacts with
Business Delegate
Caching, service consolidation, and other client-side logic
Encapsulates
SOAP Proxy
Client-side proxy for server-based EJB component
Serializes requests and communicates via
Internet/Intranet
Routes message to
SOAP Stub
Parses incoming SOAP message and calls appropriate server-side component
Deserializes request into
Transfer Object
Manages consolidated inputs and outputs from component
Presents results from
Session Facade
Presents coarse-grained component interface to server-side functions
Delegates to
Composite Entity
Assembles server-side business logic and content into loosely coupled entity beans

FIG. 6: ALL PATTERNS AND THEIR DEPENDENCIES

Q PASA! TIVOLI INTEGRATION ENHANCES MANAGEMENT OF WEBSPHERE MQ ENVIRONMENT

(Minneapolis, MN) – MQSoftware, Inc., a provider of middleware technology solutions, has announced an integrated version of its Q Pasa! middleware management system with IBM's Tivoli System Management solution. This new offering delivers tighter integration between Q Pasa! and Tivoli management software that will allow businesses to manage detailed information for IBM WebSphere MQ via powerful, specialized application capabilities. The Tivoli integration enhancement is immediately available in the latest release of Q Pasa!.

"Using Q Pasa! along with Tivoli to monitor and manage a variety of IBM integration offerings – from WebSphere Business Integration to WebSphere MQ – will help customers link a variety of business processes together, increasing efficiency across the enterprise," said Bob Madey, vice president of IBM's Tivoli Software.
www.mqsoftware.com

PARASOFT UNVEILS ENHANCED ERROR-PREVENTION TOOL

(Monrovia, CA) – Parasoft, the leading provider of automated error-prevention software solutions, previewed SOAPtest 2.0, its automated testing tool for Web services, at the Web Services Edge Conference & Expo in Boston last month.

SOAPtest's automated technologies help development teams prevent errors by performing server functional testing, load testing, and client testing with just the click of a button. In addition, developers can also use SOAPtest as a proxy server to view and verify messages between a client and a Web service.

"If Web services are really going to change the distributed computing landscape, the need for such applications to be reliable and error free is more critical than ever," said Gary Brunell, vice president of professional services for Parasoft. "SOAPtest is designed to prevent errors in Web services from the earliest stages of development."

SOAPtest 2.0 will be available May 15th from Parasoft and will run on Windows 2000/XP, Linux, and Solaris.
www.parasoft.com/soaptest

COCOBASE ENTERPRISE O/R MAPPING TOOL INTEGRATES WITH WEBSPHERE 5

(San Francisco) – The Dynamic O/R Mapping Company has announced that the new release of CocoBase Enterprise O/R version 4.5 Service Release 1 is integrated with IBM's WebSphere Application Server v5. Developers can now use the award-winning object-to-relational mapping tool, CocoBase, to build and manage database access when creating applications to run on WebSphere Application Server v5. Now available to developers is the ability to easily persist the full range of data needs, including complex object graphs and entire database schemas using CMP and BMP entity beans, session beans, and Distributed Dynamic Transparent Persistence.

CocoBase is an ideal solution for the entire IBM Java tools platform, according to Thought Inc., as it is also tightly integrated with WebSphere Studio/Eclipse/VisualAge development environments, and the Rational Rose UML modeling tool. The combination of these best-of-breed tools with CocoBase's Dynamic Object to Relational Mapping architecture greatly simplifies the development process and cuts the cost of database access development by up to 85%, according to the company.

CocoBase can be downloaded for free from the company's Web site.
www.thoughtinc.com

ADONIX BASES NEW ERP/E-BUSINESS SOLUTION ON WEBSPHERE - EXPRESS

(Pittsburgh) – Adonix has announced that it will bring its Adonix X3 ERP software suite to IBM's WebSphere - Express platform. Adonix will provide midmarket companies with a robust solution – based on WebSphere's market-leading, open standards–based, e-business software – that is packaged, affordable, and easy to implement.

"By teaming with IBM, we can now provide midsize companies with a complete e-business enabling solution," said Richard Burns, vice president of development for Adonix. "This partnership will create an "open" ERP environment that combines Web deployment tools and enterprise-class security from IBM with Adonix X3's unique Internet architecture, making it an ideal solution for leveraging the Web to better serve customers."

Adonix X3 is an easy-to-implement, Web-based ERP solution that gives a new level of power and flexibility to mid-market enterprises without the cost and risk typically associated with such systems, according to the company.
www.adonix.com

DELOITTE CONSULTING TO OFFER NEW SOLUTIONS BASED ON IBM TECHNOLOGY

(New York) – Deloitte Consulting, soon to be Braxton, has announced that it will offer RealTime Powered by WebSphere and Deloitte Consulting's Collaborative Commerce Solution to allow companies to execute finely tuned value chains, leading to higher profits and revenue growth.

By providing the e-business on demand benefit of seamless integration of business events and processes within an organization, RealTime Powered by WebSphere can help organizations gain a competitive advantage, reduce overhead, and increase profits, the company says.

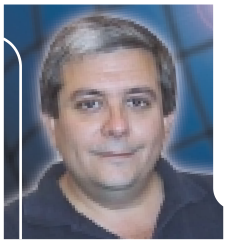
"Companies today are facing two critical challenges," says Bob Dalton, principal with Deloitte Consulting. "First, they do not have the time or money to support long-term projects and instead need rapid ROI on their performance improvement initiatives. At the same time, they need to extend their business beyond the four walls of their organization and work collaboratively with their ecosystem of trade partners.

RealTime Powered by WebSphere integrates WebSphere Portal, WebSphere Application Server, and WebSphere Business Integration to enable critical integration services such as user interaction, business processes, data integration, application connection within an enterprise, and communication between trading partners.

The products and free 30-day evaluation may be downloaded from the company's Web site.
www.dc.com.

On IT's Uncertain Future

BY JIM MARTIN



A few months back, in this very spot, I wrote that in 2003 we would see an uptick in IT spending. Then I asked the question, "Where will the money be spent?" Some read this as welcome news foreshadowing the revitalization of the IT sector; others viewed it as wishful thinking. Clearly those who see it as wishful thinking also see the glass as half empty. These people are running out of what made IT grow and flourish in the first place: innovation.

It's still a little too early in the year to declare which camp is correct. However, many signs point to new IT spending on both systems and software. Recent studies by Aberdeen Group, IDC, and Yankee Group have all forecast conservative but steady single-digit IT growth for 2003. This isn't enough to make us throw wild, vintage dot-com parties, but it's better news than we've heard in quite a while. I believe the worst is behind us now, but the future is uncertain for two reasons: first, the whole economy can change very quickly with the winds of war. Second, the new IT spending isn't doing anything to help American technology companies.

Now that American investors have been blown away by the recent accounting scandals, how do corporations restore the bruised and battered bottom line? IT companies are finding the answer under the same rock that manufacturing companies looked under: reduce the cost of the labor component in the finished product.

If you look back over history, this has been a tried-and-true formula for many industries born in the United States: textiles, heavy machinery, automobiles, and electronics have all been sent to the lowest-cost labor market overseas to return to our shores with their only American-made component being a brand name. In some cases the quality remained the same; in some cases it even improved, but in most cases we got something less than we had before. The same can now be said for IT.

For a moment let's put the quality of deliverables to the side; let's look at basic economic fundamentals. Domestic spending is the engine that moves the economy. IT spending spurs the national economy and to an unhealthy degree influences the stock market. The increased IT spending that we are seeing in 2003 accomplishes neither of those things because the lion's share of the spending is not being spent in the United States; it's not helping put American IT workers back onto payrolls; and it definitely is not a catalyst for innovation. Remember when Ross Perot talked about manufacturing and NAFTA (the North American Free Trade Agreement), saying you would hear a giant sucking sound heading south? He was right. Now we hear a giant sucking sound heading east, but this time it's IT leaving our shores. Foreign IT spending does very little to help our economy and does not foster continued innovation in software. All it does is reduce the cost of the labor component of deliverables. While this is an effective short-term measure to bolster earnings or in some cases a last-ditch effort to stave off the inevitable, the long-term effects of this practice will only harm our economy.

Offshore IT resources cannot be ignored. They do provide a lower-cost environment, and in many cases an equally good deliverable. However, other factors have to be examined. How does it affect America's place within the global economy? It only weakens us because American IT workers are not benefiting. We have less money to spend, which weakens our national economy, and as our skill sets are supplanted by offshore workers we lose our incentive and means to innovate – and in many cases we even put our technology resources at risk. In a global economy, risk needs to be evaluated more precisely than ever before. War, terrorism, internal discord, and even natural disasters need to be weighed long term, not just on a quarter-by-quarter basis.

I know that this smacks of nationalism and protectionist policy, but we do need to look out for America first. If we don't, who will? After all, American history is the story of creating, perfecting, and bringing new things to market. Remember Henry Ford? He said he wanted to build a car that his employees could afford. Why? He would build and sell more cars and thus make more money. As he made more cars he added more workers who had money to spend on his cars and other things, which helped our economy grow. How does this translate to the modern IT age? Out-of-work IT workers don't buy new PCs or upgrade software. They don't buy much of anything, which stifles our economy. As American IT workers adjust and migrate to other sectors, we will lose our ability to be innovative and first to market with the next wave of technology.

Gone are the modern, upscale, overpriced offices of the dot-com bust. Gone are the huge trade shows. And gone are the days of shooting fish in the IT barrel. Now we need to be competitive and brilliant. We'll lose that ability if we stay focused only on this quarter's bottom line; we need long-term vision. So, dear reader, I ask you to consider where you want America and its IT sector to go in the future. After all, if you're reading this, your future is probably tied to the future of American IT.

ABOUT THE AUTHOR... Jim Martin has worked in the system integration and communications industry for the past 15 years. Working on design and implementation teams, he has been instrumental in deploying Web-based mission-critical systems.

E-MAIL
jmartin@csstrategies.com

QUEST SOFTWARE

HTTP://JAVA.QUEST.COM/PERFORMANCE/WDJ

TRILOG GROUP

WWW.FLOWBUILDER.COM