SAPinsider

This article appeared in the Oct ▪ Nov ▪ Dec 2003 issue of *SAP Insider*. It appears here with the permission of WIS, publisher of *SAP Insider*. Visit **www.WISpubs.com** for additional resources.

Web Services for Platform-Independent, Standards-Based Information Sharing

by Martin Huvar, SAP AG

Opening up your business processes in a heterogeneous IT landscape is often a painful exercise, due to such hurdles as non-standardized or proprietary communication and messaging systems, platform dependencies, and security issues. Technologies like EDI, COM/DCOM, and CORBA have tried to address these problems, but they fall short in other areas such as specific platform requirements, client/server dependencies, or high total cost of ownership. Based on open and commonly accepted standards, Web services promise to solve these problems.

Web services provide a standardized, platform-independent way for applications and organizations to share information and functionality. What's more, Web services can lower the costs of maintaining and integrating processes that span various platforms and solutions, including those of your partners and customers. So, how can you use Web services to get more out of your IT investments?

Start with SAP NetWeaver Developer Studio, SAP's integrated development environment and tools. It not only supports Java, J2EE, and Web Dynpro development,[1] but it also provides the tools and wizards for easily creating, reusing, and modifying Web services — sometimes in just a few mouse clicks. SAP offers a flexible, extensible Web Services Framework as part of their standard development processes, to ensure that Web services can be implemented with the same mechanisms and tools, and that they use the same services (transport landscape, Java Development Infrastructure,[2] etc.) as other SAP applications.

This article suggests some areas where you should consider using Web services. Once you've selected the business process and appropriate interfaces to expose as Web services, you'll find tools to guide you through the basic steps within SAP NetWeaver Developer Studio. You'll also find client-side support to help you incorporate new cross-platform functionality into your current business processes.

Providing Valuable (Web) Services to the World — The Server Side

Any Web service starts with the business process itself and identified, standard implementations of business functionality. These can range from credit-card checks and currency-exchange computations to payroll functions for your employees.

Based on the standard interfaces of a business application, any self-contained, modularized functionality that is implemented either as an Enterprise JavaBean (session bean) or as a Java class lends itself to deployment as a Web service. This functionality could be provided by SAP as part of a mySAP solution or developed by you, your customer, or your partner. This section takes a look at how you would create and register the Web service on the server side.

SAP NetWeaver Developer Studio offers two options for building a Web service: one is a quick wizard-based approach; the other is more involved, but provides the flexibility for custom-designing more complex Web services.

> ✔ **Note!**
> These two approaches to creating a Web service aren't necessarily mutually exclusive. You can quickly create the Web service with the wizard and return later to manually fine-tune the default settings using the second, more advanced approach.

[1] See the articles by Karl Kessler and Peter Tillert in this issue of *SAP Insider* (www.SAPinsider.com).

[2] Want to know more about the Java Development Infrastructure (JDI)? Have a look at the article by Wolf Hengevoss in this issue of *SAP Insider*.

Creating a Web Service in Less than a Minute

Once you've identified a process and an interface that you want to make available, the easiest way to start is with SAP NetWeaver Developer and the Web services creation wizard (see **Figure 1**).

With this wizard, configuring a Web service is a highly automated process based on predefined configuration profiles developed by SAP that bundle settings used in typical Web services scenarios.

The configuration profile, *Simple SOAP*, for example, is set for stateless Web service calls over an intranet, where no additional security is required. In Figure 1, this profile includes no authentication or authorization functions and specifies the use of SOAP over HTTP for transport (see the sidebar "Web Service Standards"). Additional profiles combine other settings, including basic authentication, strong security and authorization, and so on.

With these wizards, even a Java developer who is less experienced with detailed technical aspects of the Web service can still create one simply by clicking through the three (yes, three!) steps of the wizard (see "Who Creates the Web Service?" in the online version of this article at **www.SAPinsider.com**). In the background, all the required objects (discussed in the next section) are generated to the appropriate projects.

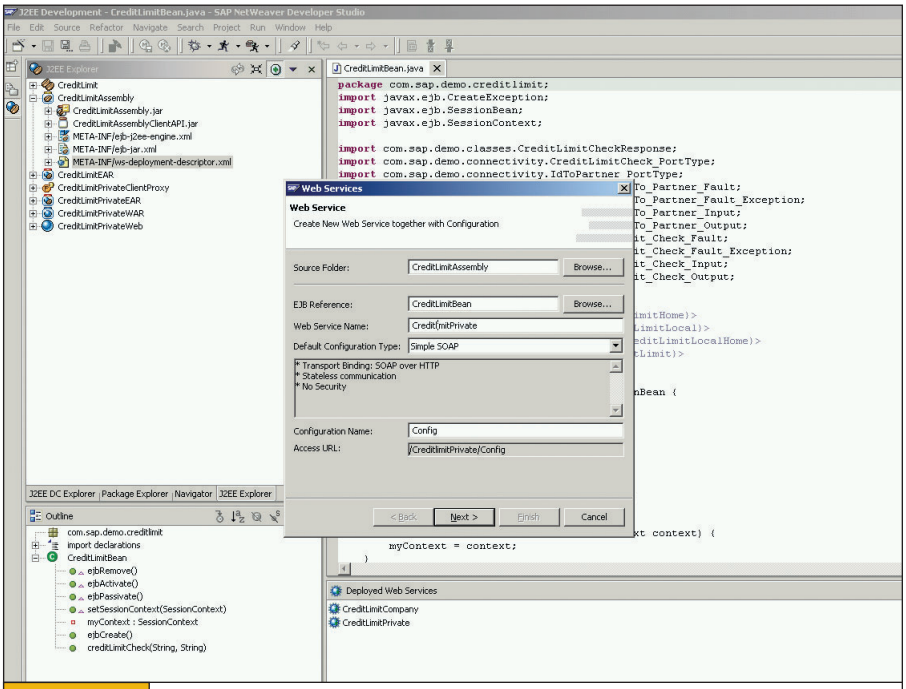

Figure 1 Wizard for Creating the *CreditLimitPrivate* Web Service

What Are Web Services?

The term "Web services" here refers to *application functionalities* that support *direct interaction* by responding to *service requests* based on *open Internet standards*.

So how does this apply to *your* business processes and integration efforts?

Suppose employees throughout your company routinely perform credit checks on business partners, verifying their credit limits by phone or fax. This is the kind of consistent business process that lends itself nicely to standardization and componentization. Automating this process requires open integration across various client platforms — a perfect candidate for Web services.

With Web services in place, employees use the service directly from their business application and request information via a portal application or a Web form. This request invokes the corresponding Web service at the appropriate credit agency or at a Web services-enabled backend system located inside or outside the company. The results from this request can also be easily used in follow-up business processes.

Web services are often an ideal choice for opening up your business processes because they:

- Can be discovered, described, and called over standard Web protocols.
- Act like a "black box" — they can accept input and deliver a result.
- Are modular, self-contained, and self-describing.
- Work on top of any communication technology stack.
- Can be published, discovered, and invoked based on open technology standards.
- Work in synchronous and asynchronous scenarios.
- Facilitate integration within an enterprise as well as across enterprises.
- Can be utilized by user interfaces for interactive scenarios.

The Step-by-Step Approach for More Complex or Custom Web Services

There are times when ease and speed of development will take a back seat to more advanced processes in order to meet special requirements. **Figure 2** shows a five-step process for exposing a business application as a Web service without the help of the creation wizard.

Step 1: Implement the Business Logic

If you can implement a Java class or enterprise session bean, you can implement the business logic for a Web service. It doesn't involve anything specific to Web services — you simply implement a Java class or follow the enterprise session bean model. Both approaches can be used for building Web services and are supported from either the Java or J2EE perspectives in SAP NetWeaver Developer Studio.[3]

Step 2: Define the Virtual Interface

The virtual interface (VI) is the interface that your Web service will expose to clients (i.e., consumers). As its name suggests, the virtual interface is abstracted from the "actual," original, business application interface.

The default for the VI is a 1:1 mapping between the original, implemented interface and the interface exposed to the client. You're then free to tailor this default interface, using a VI editor provided by the NetWeaver Developer Studio, to:

- Rename methods and parameters
- Hide methods or parameters
- Provide default values for parameters
- Decide whether parameters are treated as attributes or elements in the SOAP message
- Change namespaces
- Map basic data types according to Java standards

This means that you can easily define any number of views on top of the original, implemented interface, in order to deliver specially tailored, platform-independent interfaces to Web services consumers.

[3] For more on the perspectives available in the NetWeaver Developer Studio, see Karl Kessler's article in this issue of *SAP Insider* (www.SAPinsider.com).

Web Services Standards: WSDL, SOAP, and UDDI

Web services offer a model of plug-and-play, peer-to-peer, cross-platform collaboration. This presupposes that well-known and commonly accepted standards for Web services are utilized, which include:

☑ **WSDL (Web Services Description Language)**
This is a vendor- and platform-independent description of Web services that allows for a straightforward, quick development process. WSDL includes the description of the Web service interface using XML schema.

☑ **SOAP**
This protocol describes how to invoke a Web service described by a WSDL document. SOAP specifies an envelope for exchanging XML documents, an appropriate error-handling mechanism, and the binding to a specific transport protocol like HTTP.

☑ **UDDI (Universal Description Discovery and Integration)**
An optional, but useful, building block in this process, UDDI is a place to publish and retrieve Web services. A UDDI registry holds metadata that allows you to list and search for a Web service based on names, IDs, categories, types, etc. (See the sidebar "Full Support for UDDI Registration and Retrieval," on page 5.)

Other standards are already established or are in the works to help solve Web services challenges such as security, the choreography of messages between Web services, or standardization of business documents and processes.

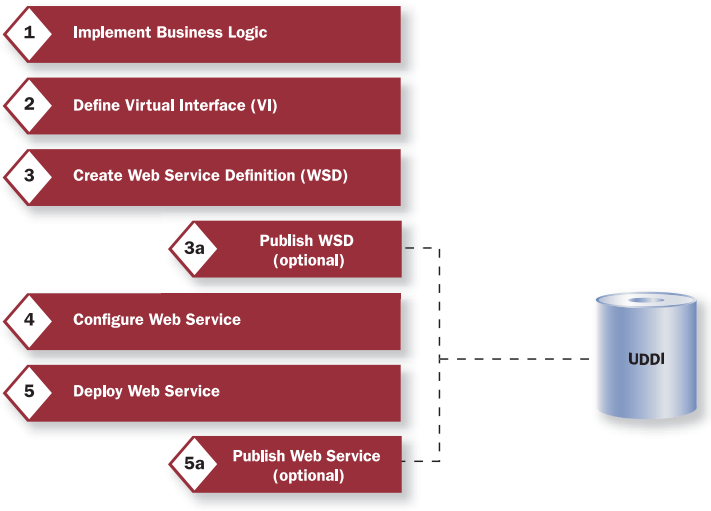

Figure 2 Five Easy Steps to Deploying a Web Service with the SAP Web Services Framework

Step 3: Create the Web Service Definition

Based on the virtual interface, the developer creates a Web services definition using a wizard. This step defines the Web service's behavior on a general level. Supported by an editor tool, session handling, authentication, authorization, and transport guarantees are all defined in this step.

Step 4: Set Up the Web Service Configuration

In your system landscape, different application servers may have different technical capabilities. In this step, the general features defined in Step 3 are mapped to the actual technical capabilities of the server. So if the developer indicated that this Web service requires stateful communication in Step 3, the IT team[4] can determine at this stage whether this should be realized using HTTP cookies or URL extensions.

Step 5: Deploy the Web Service

Java developers will be pleased to learn that deployment involves no special Web services-specific tasks. Deployment processes are fully integrated into the Developer Studio, and the entire configuration for deployment is handled transparently in the background.

Testing Capabilities — Quick Turnaround Times in Development

After creating a new Web service, you'll want to test it out. So for every deployed Web service, a Web service homepage is also automatically created and deployed (see **Figure 3**).

Using a browser-based interface, the homepage bundles the complete documentation, shows the associated WSDL files, allows you to generate client proxies, and offers testing capabilities. You can check every Web service without any additional coding to implement a test client.

A full status overview completes the homepage, showing selected features including the status of UDDI publishing.

[4] For more on division of developer tasks in this 5-step process, see the section "Who Creates the Web Service?" in the online version of this article at www.SAPinsider.com.

Adding Web Services to Your Business Processes — The Client Side

The Web Application Server not only supports the creation of Web services on the server side; calling Web services as a Web service client (or consumer) is the second role the SAP Web Application Server can play. And just as on the server side, the developer will find plenty of support from SAP NetWeaver Developer Studio[5] on the client side.

There are four basic steps for running a Web service application from the client side:

Step 1: Retrieve the Web Service Description

A standards-compliant Web service description is the starting point for implementing a client application. To find a suitable Web service, either browse the UDDI with the integrated Web services browsing tool or add your own WSDL description manually.

[5] The developer on the Web service client side will be familiar with the Web services programming model, client proxies, etc. See "Who Implements the Web Service?" in the online version of this article at www.SAPinsider.com.

Step 2: Generate a Web Service Proxy

Using a valid WSDL file as input, a Web services client proxy is generated. The proxy allows the application developer to focus on business functionality, while technical aspects like creating a valid SOAP message are automatically done by the proxy implementation.

Step 3: Implement the Client Application

Different types of implementations can use the generated Web service proxy. Depending on the needs of the project, you might choose Enterprise JavaBeans, Java classes, or Java Server Pages. Whichever is used, the implementation of the client functionality follows the standardized and commonly accepted Web service programming model.

Step 4: Deploy the Application

Last but not least, the deployment needs to be executed. As on the server side, deployment is based on integration with standard SAP Web AS processes, so no Web services-specific actions are required.

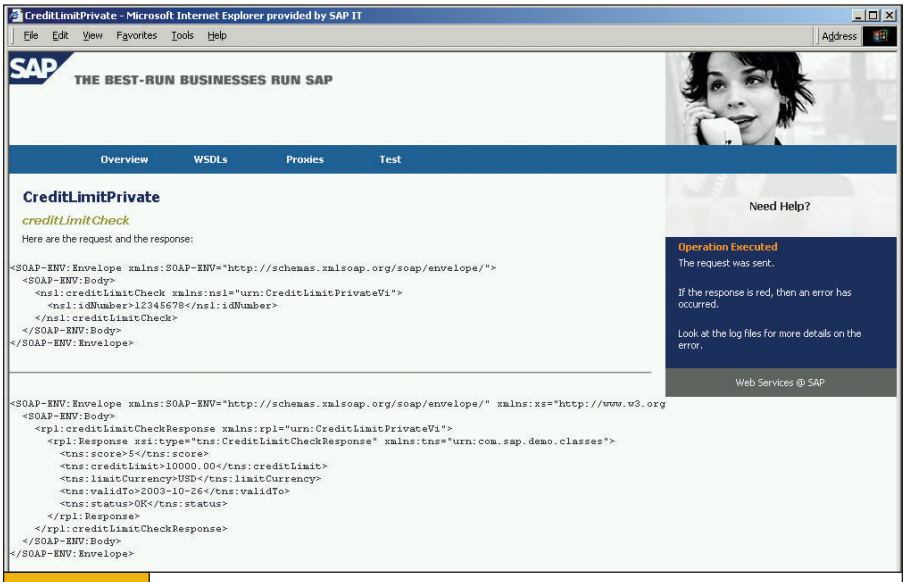

Figure 3 Homepage for the *CreditLimitPrivate* Web Service

Full Support for UDDI Registration and Retrieval

The SAP Web Services Framework offers full-fledged UDDI client functionality based on UDDI Specification 2.0. This allows for browsing, querying, and retrieving Web services and Web service types based on standard UDDI APIs. This also allows you to publish Web services and Web service definitions to any UDDI-compliant registry (see Steps 3 and 5 detailed earlier in this article).

In addition, a UDDI server is shipped with SAP Web Application Server 6.30. Fully integrated into the standard administration and with plug-and-play installation, UDDI server capabilities are added into your system landscape, so every UDDI-compliant client can interact with your UDDI server.

Conclusion

With SAP Web Application Server 6.30, SAP offers an easy, convenient way to build Web services. SAP's rich business functionality, in conjunction with state-of-the-art technology, enables you to establish cross-company business processes as an integrated part of your development efforts.

Anticipating the proliferation of comprehensive business processes based on Web services, SAP has introduced the Enterprise Service Architecture (ESA), a newly developed approach for building services-oriented business applications. ESA offers a blueprint for creating new, flexible, modular applications that support communication in heterogeneous environments. SAP NetWeaver and its application platform, SAP Web Application Server, are the foundation for all mySAP solutions, and the Web services capabilities of SAP Web AS are a vital part of the ESA strategy.

As Web services continue to evolve, new standards will further enhance security, support more complex messaging among multiple Web services, and lead to even greater standardization of business documents and processes. SAP is one of the leading companies driving Web services technology and business standards and fostering greater openness overall in business solutions and processes. This support encourages wide adoption of Web services and ensures the continued evolution of supporting technologies.

Flexibility, manageability, and reliability of heterogeneous IT environments at low operational costs are key factors for future company success. Web services provide for integration of business functions within and across enterprises and on top of any communication technology stack — and, ultimately, help to safeguard your technology investments.

For more information, see **www.sap.com/solutions/netweaver**. ■

Martin Huvar joined SAP about ten years ago. After some years of development and consulting in different technology areas, he worked as Product Manager for the SAP Business Connector and XML Technology. He is now the responsible Technology Product Manager for Web services at SAP, a position he has held for the past 1½ years. Martin can be reached at martin.huvar@sap.com.