✳ PROGRESS® DataDirect®

CREATE CUSTOM DRIVERS FOR DATA-DRIVEN WEB SERVICES

INTRODUCTION

The current architectural trend is to wrapper applications with Web Services in what is called a Service-Oriented Architecture (SOA). This is done in hopes of creating an enterprise architecture that is flexible to the needs of various information consumers. A similar trend is at work for ASP and eCommerce sites like SalesForce, eBay, and Amazon. They want to enable customization and interoperability with their platforms through Web Services.

This means the functionality contained within these applications is now accessible to applications that can support Web Services requests. But most of the desktop applications used by business users run as Windows or UNIX applications that do not have built in support for accessing services through SOAP. There is a need to migrate to SOA but at the same time there is a demand from the users to maintain compatibility with the hundreds of applications they use for reporting and analysis.

The use of Progress® DataDirect® OpenAccess™ to implement a custom ODBC or JDBC driver with full SQL capability is the best solution for bridging the current applications and the Web Services worlds. Most of today's reporting, analysis, and database applications have the ability to interface to data sources through ODBC or JDBC. With OpenAccess you can leverage this widespread feature to enable your Web Services compliant application to be accessed—without requiring clients to change their applications.

HOW TO QUICKLY IMPLEMENT A CUSTOM JDBC/ODBC DRIVER OVER WEB SERVICES

OpenAccess SDK provides the framework and pre-built components to quickly allow the implementation of a custom JDBC and ODBC driver for any data source that has a Web Services interface (Figure 1).

HIGHLIGHTS:

▶ Up and running in just over one week

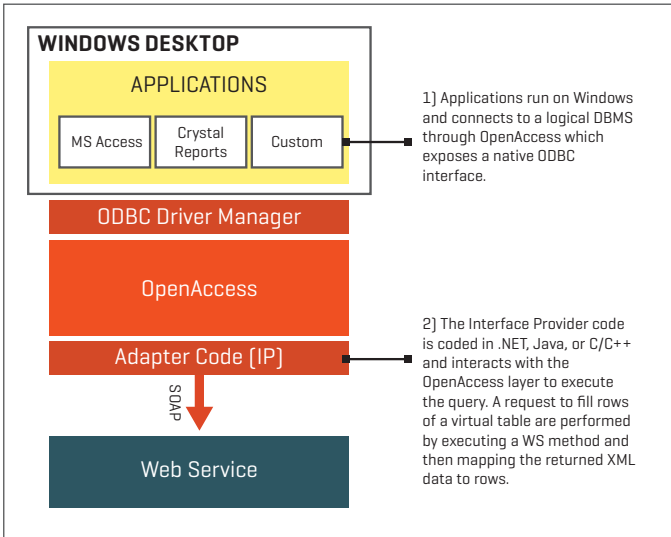▶ Leverage ODBC or JDBC to enable web services compliant applications

1) Applications run on Windows and connects to a logical DBMS through OpenAccess which exposes a native ODBC interface.

2) The Interface Provider code is coded in .NET, Java, or C/C++ and interacts with the OpenAccess layer to execute the query. A request to fill rows of a virtual table are performed by executing a WS method and then mapping the returned XML data to rows.

Figure 1: *OpenAccess Based Solution*

✳ PROGRESS

For example, to ODBC-enable Web Services like the ones exposed by SalesForce.com or Amazon.com, the steps are:

1. Obtain the WSDL

2. Define how this service will be exposed in terms of virtual tables

3. Use a tool like Visual Studio.NET to generate wrapper classes to access the Web Services methods

4. Implement glue code (< 500 lines of code) to tie in this class to the OpenAccess SQL engine to support schema and data access requests. Assuming you are exposing a virtual table called ACCOUNTS, a user can perform a query like:

```
select * from ACCOUNTS where balance > 3000
```

which would be handled through the use of the custom Interface Provider (IP) code to make WS method calls, which returns XML formatted data for ACCOUNTS and then maps the XML formatted data into rows.

YOUR DEVELOPMENT EFFORT

1. Design and code the adapter code in either C, C++, Java, or .NET (5 days)

2. Do your QA (2 days)

3. Package up for distribution (1 day)

   Expected time of completion: **8 man days**

CONCLUSION

The ability to expose a Web Service as a virtual SQL database with ODBC and JDBC allows hundreds of Windows and UNIX applications to leverage the investment being made in SOA. With OpenAccess, you can quickly implement an enterprise quality ODBC or JDBC driver because it includes 99% of what is needed to implement a driver. The remaining 1% is specific to the Web Service and can be coded in Java or .NET in a very short time because both of these environments provide rich support for Web Service access and XML processing.

www.progress.com

✳ PROGRESS