What's Brewing With Java

Java is the foundation of our next generation product, BBj®. Here, we track and analyze the developments taking place in the world of Java. In this issue, BASIS Software Engineer Jeff Ash explains some basics about Java Database Connectivity.

More Data Access Choices: JDBC

By Jeff Ash

When information technology professionals are faced with a business requirement, it is important that they have as many technology choices as possible. Prior to the release of BBj data access components, developers and users had only two choices available for accessing BBx data: BBx applications and Open Database Connectivity (ODBC). BBj adds a third option for accessing data: JDBC (Java Database Connectivity).

What Is JDBC?

JDBC is a Java application programming interface (API) that provides a method of accessing data that is both platform independent and database management system (DMBS) independent. JDBC is similar in function to ODBC, a common interface used with Microsoft Windows applications. Just as ODBC can be used to access data from a Windows application, the JDBC API can be used to access data from a Java program or from an off-the-shelf Java application, such as a Java-based report writer. If you're using the JDBC API from your Java program, you'll find that its object-oriented structure is easy to use. The API is also supported by a large number of Windows- and Unix-based third-party products.

How JDBC Differs From ODBC

The first thing to understand about JDBC is that it is not a replacement for ODBC but rather an additional technology for accessing data. JDBC is an API used through the Java language, while ODBC is a native interface accessed through a language that can make calls into a native library (for example, DLL). JDBC is a huge advantage because it provides platform-independent access to any database that has a JDBC driver available.

Another difference between JDBC and ODBC is that JDBC is object-oriented and does not use pointers. This makes for a much cleaner interface that is easier to use. In this way, JDBC closely resembles some of the ODBC-wrapper APIs, such as ADO and DAO.

JDBC Programming

Developing applications using JDBC is much simpler than using ODBC directly or other database APIs. JDBC consists of a package of Java classes and interfaces that

encapsulate the various aspects of data access. The following is a brief description of the basic classes used in a JDBC application:

- Driver – Each DBMS must have a JDBC driver specific to that system for the DMBS to be accessed by a JDBC application. The driver class is responsible for creating a connection to a particular database.
- Connection – A connection object represents a single JDBC connection to a database.
- Statement – Statement objects represent a single SQL statement. Statement objects can be queries, updates or data-definition language statements.
- ResultSet – ResultSet objects represent the results of an SQL query.
- Meta Data – The various meta data classes provide methods for retrieving certain information about databases or result sets. The DatabaseMetaData class gives the user the ability to get information about the database itself, such as available tables, indexes, version, etc. The ResultSetMetaData class provides methods for retrieving information about ResultSet objects, such as columns returned in a query, data types of query columns, etc.

The following sample code shows the simplicity and object-oriented nature of the JDBC API. This code makes a connection to the local BBj Data Server's Chile Company database, executes a statement and prints the contents of the ResultSet to standard out:

```
import java.sql.*;

public class JavaSample
{
    public void main(String[] p_args)
    {
      // Setup the connection string (URL)
        String url = "jdbc:basis:localhost?DATABASE=ChileCompany";

        // Load the BASIS JDBC Driver into the process so we can use it
        Class.forName("com.basis.jdbc.BasisDriver");

        // Make a connection to the Chile Company database
        Connection con =
          DriverManager.getConnection(url, "theuser", "thepwd");

        // Create a statement and execute it.
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT item_num FROM item");

        // Show the results
        while (rs.next())
        {
            System.out.println(rs.getString("ITEM_NUM"));
        }

        rs.close();
        stmt.close();
        con.close();
    }
}
```

The powerful JDBC API creates another avenue by which BBj data can be accessed by third-party products, such as report writers, application servers and Web-based applications. Use of the BBj Data Server and its platform-independent JDBC driver open up a whole new world of technologies to the Business Basic community.

Additional Information

Check out Sun Microsystems' JavaSoft Web site at

You'll find a complete description of the JDBC API, documentation and easy-to-follow tutorials.