Servoy's Database Connectivity

At its core, Servoy is a comprehensive platform to develop database-driven applications. As such, Servoy allows developers to connect to any standard Relational Database Management System (RDBMS). To achieve this, Servoy employs Java Database Connectivity (JDBC) technology. JDBC is a connectivity standard that allows Java-based applications to interact transparently with any database vendor that provides a compliant driver

In This Chapter

Query Engine

The Servoy platform uses Structured Query Language (SQL), a standard communications protocol to issue requests to databases. The traditional development of database applications typically requires advanced SQL knowledge to adequately and efficiently retrieve data. Moreover, database vendors often adapt their own "flavors" of SQL, making database portability an issue. If an application uses non-standard SQL expressions, it cannot be easily deployed against databases other the the one for which it was developed.

The Servoy platform obviates the need for developers to write their own SQL in almost all scenarios. Instead Servoy dynamically generates all the SQL required to read and write data for all but the most complex scenarios. Servoy's generated queries are guaranteed to be optimized and database-neutral. At the same time, the Servoy platform is open and allows developers with the knowledge and preference for writing SQL to do so as well.

Named Connections

Developers will specify a *Server Name*, which maps to a specific database connection configuration (i.e. user, password, URL, etc). At run time, Servoy will automatically create a pool of connections for a given Server Name, which will be reused for the duration of the application server up time. Developers are always insulated from the complexities of connecting to the database.

The details of a connection configuration are stored as part of the development or deployment environment and are never part of the code base. This allows the database connection to be changed for a given context without making modifications to an application's code base.

For example, it is common to have separate databases for development/testing and production. Therefore, a Server Name could resolve to a test database in both Servoy Developer and an instance of Servoy Server used for staging. The same Server Name would resolve to a production database for an instance of Server Server used in production. Another example is for multiple, on-premise, deployments where local instances of Servoy Server rely on local database connections.

Switching Connections

While database connections can be changed transparently between different development and deployment contexts, they can also be changed within the same context. Servoy's API provides a means for developers to change, at run time, from one Server Name to another. For example, different application user groups may be required to use different connections to the same database. In another example, different customers may have their data stored in their own separate database. In both these cases, a specific Server Name is identified and used for an individual client session.

Connection Pooling

Servoy uses database connection pooling technology which provides significant benefits in terms of application performance, concurrency and scalability. Database connections are often expensive to create because of the overhead of establishing a network connection and initializing a database connection session in the back end database. Moreover, the ongoing management of all of a database's connection sessions can impose a major limiting factor on the scalability of an application. Valuable database resources such as locks, memory, cursors, transaction logs, statement handles and temporary tables all tend to increase based on the number of concurrent connection sessions. This limitation is overcome using Connection Pooling, whereby a limited number of connections is shared by a larger number of clients. When a client makes a new request for data, the application server briefly borrows a connection from the pool to issue the query, then returns it to the pool. In this manner an application can scale well beyond the limits of the database without compromising performance. Servoy's connection pools are configurable so that connectivity can be optimized to suit applications of various sizes.