3
Configuring JDBC Data Sources

In WebLogic Server, you configure database connectivity by adding JDBC data sources to your WebLogic domain. Configuring data sources requires several steps including choosing a type of data source, creating the data source, configuring connection pools and Oracle database parameters and so on.
This chapter describes the steps required to create and configure JDBC connection pools. It includes the following topics:

Understanding JDBC Data Sources

In WebLogic Server, you configure database connectivity by adding data sources to your WebLogic domain. WebLogic JDBC data sources provide database access and database connection management.
Each data source contains a pool of database connections that are created when the data source is created and at server startup. Applications reserve a database connection from the data source by looking up the data source on the JNDI tree or in the local application context and then calling `getConnection()`. When finished with the connection, the application should call `connection.close()` as early as possible, which returns the database connection to the pool for other applications to use.

Types of WebLogic Server JDBC Data Sources

WebLogic Server provides five types of data sources such as Generic data source, Active GridLink (AGL) data source, Multi data source (MDS), Proxy data source and Universal Connection Pool (UCP) data source.

- Generic data sources—Generic data sources and their connection pools provide connection management processes that help keep your system running efficiently.You can set options in the data source to suit your applications and your environment.
- Active GridLink (AGL) data sources—A datasource that provides a connection pool that spans one or more nodes in one or more Oracle RAC clusters. It

8

Using Universal Connection Pool Data Sources

5

A Universal Connection Pooling (UCP) data source is provided as an option for users who wish to use Oracle Universal Connection Pooling to connect to Oracle Databases. UCP provides an alternative connection pooling technology to Oracle WebLogic Server connection pooling.

This chapter provides information on how to configure and monitor UCP data sources. This chapter includes the following topics:

What is a Universal Connection Pool Data Source?

A Universal Connection Pooling (UCP) data source is provided as an option for users who wish to use UCP for connecting to Oracle Databases. UCP provides an alternative connection pooling technology to Oracle WebLogic Server connection pooling.

> **Note:**
>
> Oracle generally recommends the use of Active GridLink data source, multi data source, or JDBC data source with Oracle WebLogic Server to establish connectivity with Oracle databases.

WebLogic Server provides the following support when using a UCP data source:

- Configuration as an alternative data source to Generic data source, Multi Data Source, or Active GridLink data source.
- Deploy and undeploy data source.
- Basic monitoring and statistics:
  - `ConnectionsTotalCount`
  - `CurrCapacity`
  - `FailedReserveRequestCount`
  - `ActiveConnectionsHighCount`

Here is an example:

```
...
OracleDataSource odsconn = (OracleDataSource)ctx.lookup("jdbc/sampledb");
Connection conn = odsconn.getConnection();
...
```

Logging and Tracing

The data source facility offers a way to register a character stream for JDBC to use as output for error logging and tracing information. This facility allows tracing specific to a particular data source instance. If you want all data source instances to use the same character stream, then you must register the stream with each data source instance individually.

The `OracleDataSource` class implements the following standard data source methods for logging and tracing:

- `public synchronized void setLogWriter(PrintWriter pw)`
- `public synchronized PrintWriter getLogWriter()`

The `PrintWriter` class is in the standard `java.io` package.

Notes:

- When a data source instance is created, logging is disabled by default (the log stream name is initially null).
- Messages written to a log stream registered to a data source instance are not written to the log stream normally maintained by `DriverManager`.
- An `OracleDataSource` instance obtained from a JNDI name lookup will not have its `PrinterWriter` set, even if the `PrintWriter` was set when a data source instance was first bound to this JNDI name.

Connection Pooling

5

Connection pooling in the JDBC 2.0 extension API is a framework for caching database connections. This allows reuse of physical connections and reduced overhead for your application. Connection pooling functionality minimizes expensive operations in the creation and closing of sessions.

The following are central concepts:

- *Connection pool data sources*--similar in concept and functionality to the data sources described previously, but with methods to return *pooled connection* instances, instead of normal connection instances.
- *Pooled connections*--a pooled connection instance represents a single physical connection to a database, remaining open during use by a series of *logical connection instances*.

  A logical connection instance is a simple connection instance (such as a standard `Connection` instance or an `OracleConnection` instance) returned by a pooled connection instance. Each logical connection instance acts as a temporary handle to the physical connection represented by the pooled connection instance.

For connection pooling information specific to OCI drivers, see "OCI Driver Connection Pooling". For further introductory and general information about connection pooling, refer to the Sun Microsystems specification for the