8
Using the Connection Pool Manager

The following sections are included in this chapter:

8.1 Overview of Using the UCP Manager

The Universal Connection Pool (UCP) manager creates and maintains UCP instances. A pool instance is registered with the pool manager every time a new pool is created. This section covers the following topics:

8.1.1 About Connection Pool Manager

Applications use a connection pool manager to explicitly create and manage UCP JDBC connection pools. Applications use the manager because it offers full life cycle control, such as creating, starting, stopping, and destroying a connection pool. Applications also use the manager to perform routine maintenance on the connection pool, such as refreshing, recycling, and purging connections in a pool. Lastly, applications use the connection pool manager because it offers a centralized integration point for administrative tools and consoles.

8.1.2 Creating a Connection Pool Manager for UCP

A connection pool manager is an instance of the `UniversalConnectionPoolManager` interface, which is located in the `oracle.ucp.admin` package. The manager is a Singleton instance that is used to manage multiple connection pools per JVM. The interface includes methods for interacting with a connection pool manager. UCP includes an implementation that is used to get a connection pool manager instance. The following example demonstrates creating a connection pool manager instance using the implementation:

```
UniversalConnectionPoolManager mgr = UniversalConnectionPoolManagerImpl.
getUniversalConnectionPoolManager();
```

8.1.3 Life Cycle States of a Connection

Applications use the connection pool manager to explicitly control the life cycle of connection pools. The manager is used to create, start, stop, and destroy connection pools. Life cycle methods are included as part of the `UniversalConnectionPoolManager` interface.

Understanding Life Cycle States

The life cycle states of a connection pool affects what manager operations can be performed on a connection pool. Applications that explicitly control the life cycle of a pool must ensure that the manager's operations are used only when the pool is in an appropriate state. Life cycle constraints are discussed throughout this section.

The following list describes the life cycle states of a pool:

- Starting : Indicates that the connection pool's start method has been called and it is in the process of starting up.
- Running : Indicates that the connection pool has been started and is ready to give out connections.
- Stopping : Indicates that the connection pool is in the process of stopping.
- Stopped : Indicates that the connection pool is stopped.
- Failed : Indicates that the connection pool has encountered failures during starting, stopping, or execution.

8.1.3.1 Creating a Connection Pool

The `CreateConnectionPool` method of the Connection Manager creates and registers a connection pool. The manager uses a connection pool adapter to create the pool and relies on a pool-enabled data source to configure the pool properties. An application must not implicitly start a connection pool before using the `createConnectionPool` method to explicitly create the same pool.

The following example demonstrates creating a connection pool instance using the manager:

```
UniversalConnectionPoolManager mgr = UniversalConnectionPoolManagerImpl.
getUniversalConnectionPoolManager();

PoolDataSource  pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionPoolName("mgr_pool");
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL("jdbc:oracle:thin:@//localhost:1521/orcl");
pds.setUser("<user>");
pds.setPassword("<password>");

mgr.createConnectionPool((UniversalConnectionPoolAdapter)pds);
```

An application does not have to use the manager to create a pool in order for the pool to be managed. A pool that is implicitly created (that is, automatically created when using a pool-enabled data source) and configured with a pool name, is automatically registered and managed by the pool manager. Oracle recommends implicit pool creation.

Pool Naming Convention

A connection pool name must be defined as part of the configuration. The pool name provides a way to refer to specific pools when interacting with the manager. A connection pool name must be unique and cannot be used by more than one connection pool. The manager throws a `pool already exists` exception if a connection pool already exists with the same name.

Compatibility with JBoss

JBoss users can use the JBoss-specific silent reload functionality by setting the `oracle.ucp.destroyOnReload` JVM system property to `true`. When the

`oracle.ucp.destroyOnReload` property is set to `true`, then the JBoss-specific behavior automatically destroys an old pool instance prior to creating a new one with the same name. If this system property is not set or set to `false`, then UCP throws a `pool already exists` exception.

8.1.3.2 Starting a Connection Pool

The manager's `startConnectionPool` method starts a connection pool using the pool name as a parameter to determine which pool to start. The pool name is defined as a pool property on a pool-enabled data source.

The following example demonstrates starting a connection pool:

```
mgr.startConnectionPool("mgr_pool");
```

An application must always create a connection pool using the manager's `createConnectionPool` method prior to starting the pool. In addition, a life cycle state exception occurs if an application attempts to start a pool that has been previously started or if the pool is in a state other than stopped or failed.

8.1.3.3 Stopping a Connection Pool

The manager's `stopConnectionPool` method stops a connection pool using the pool name as a parameter to determine which pool to stop. The pool name is defined as a pool property on the pool-enabled data source. Stopping a connection pool closes all available and borrowed connections.

The following example demonstrates stopping a connection pool:

```
mgr.stopConnectionPool("mgr_pool");
```

An application can use the manager to stop a connection pool that was started implicitly or explicitly. An error occurs if an application attempts to stop a pool that does not exist or if the pool is in a state other than started or starting.

8.1.3.4 Destroying a Connection Pool

The `destroyConnectionPool` method of the UCP Manager stops a connection pool and removes it from the connection pool manager. A pool name is used as a parameter to determine which pool to destroy. The pool name is defined as a pool property on the pool-enabled data source.

> **Caution:**
>
> You must destroy the pool objects explicitly, before they get out of scope, for example, if an object is an automatic variable and its scope is about to be ended. This is especially important when your application deals with multiple pool objects. If you do not destroy the pool objects explicitly, then they can cause leakage of resources like connections, statements, result sets, heap memory, and so on.

The following example demonstrates destroying a connection pool:

```
mgr.destroyConnectionPool("mgr_pool");
```

An application cannot start a connection pool that has been destroyed and must explicitly create and start a new connection pool.

8.1.4 Maintenance of Universal Connection Pool

Applications use the connection pool manager to perform maintenance on a connection pool. Maintenance includes refreshing, recycling, and purging a connection pool. The maintenance methods are included as part of the `UniversalConnectionPoolManager` interface.

Maintenance is typically performed to remove and replace invalid connections and ensures a high availability of valid connections. Invalid connections typically cannot be used to connect to a database but are still maintained by the pool. These connections waste system resources and directly affect a pool's maximum connection limit. Ultimately, too many invalid connections negatively affects an applications performance.

> **Note:**
>
> Applications can check whether or not a connection is valid when borrowing the connection from the pool. If an application consistently has a high number of invalid connections, additional testing should be performed to determine the cause.

Related Topics

- Overview of Validating Connections in UCP

8.1.4.1 Refreshing a Connection Pool

Refreshing a connection pool replaces every connection in the pool with a new connection. Any connections that are currently borrowed are marked for removal and refreshed after the connection is returned to the pool. The manager's `refreshConnectionPool` method refreshes a connection pool using the pool name as a parameter to determine which pool to refresh. The pool name is defined as a pool property on the pool-enabled data source.

The following example demonstrates refreshing a connection pool:

```
mgr.refreshConnectionPool("mgr_pool");
```

8.1.4.2 Recycling a Connection Pool

Recycling a connection pool replaces only invalid connection in the pool with a new connection and does not replace borrowed connections. The manager's `recycleConnectionPool` method recycles a connection pool using the pool name as a parameter to determine which pool to recycle. The pool name is defined as a pool property on the pool-enabled data source.

The `setSQLForValidateConnection` property must be set when using non-Oracle drivers. UCP uses this property to determine whether or not a connection is valid before recycling the connection.

The following example demonstrates recycling a connection pool:

```
mgr.recycleConnectionPool("mgr_pool");
```

Related Topics

- Overview of Validating Connections in UCP

8.1.4.3 Purging a Connection Pool

Purging a connection pool removes every connection (available and borrowed) from the connection pool and leaves the connection pool empty. Subsequent requests for a connection result in a new connection being created. The manager's `purgeConnectionPool` method purges a connection pool using the pool name as a parameter to determine which pool to purge. The pool name is defined as a pool property on the pool-enabled data source.

The following example demonstrates purging a connection pool:

```
mgr.purgeConnectionPool("mgr_pool");
```

> **Note:**
>
> Connection pool properties, such as `minPoolSize` and `initialPoolSize`, may not be enforced after a connection pool is purged.

8.2 Overview of JMX-Based Management in UCP

JMX (Java Management Extensions) is a Java technology that supplies tools for managing and monitoring applications, system objects, devices, service-oriented networks, and JVM (Java Virtual Machine). In JMX, a given resource is instrumented by one or more Java objects known as MBeans (Managed Beans). An MBean is composed of an MBean interface and a class. The MBean interface lists the methods for all exposed attributes and operations. The class implements this interface and provides the functionality of the instrumented resource.

The MBeans are registered in a core managed object server, known as an MBean server, which acts as a management agent and can run on most devices enabled for the Java programming language. A JMX agent consists of an MBean server, in which MBeans are registered, and a set of services for handling MBeans.

> **See Also:**
>
> - https://docs.oracle.com/javase/tutorial/jmx/mbeans/standard.html
> - *Oracle Universal Connection Pool Java API Reference*

UCP provides the following two MBeans for pool management support:

- UniversalConnectionPoolManagerMBean
- UniversalConnectionPoolMBean

> **Note:**
>
> All MBean attributes and operations are available only when the `UniversalConnectionPoolManager.isJmxEnabled` method returns `true`. The default value of this flag is `true`. This default value can be altered by calling the `UniversalConnectionPoolManager.setJmxEnabled` method. When an MBeanServer is not available, the `jmxFlag` is automatically set to `false`.

8.2.1 UniversalConnectionPoolManagerMBean

The `UniversalConnectionPoolManagerMBean` is a manager MBean that includes all the functionalities of a conventional connection pool manager. The `UniversalConnectionPoolManagerMBean` provides the following functionalities:

- Registering and unregistering pool MBeans
- Pool management operations like starting the pool, stopping the pool, refreshing the pool, and so on
- Starting and stopping DMS statistics
- Logging

8.2.2 UniversalConnectionPoolMBean

The `UniversalConnectionPoolMBean` is a pool MBean that covers dynamic configuration of pool properties and pool statistics. The `UniversalConnectionPoolMBean` provides the following functionalities:

- Configuring pool property attributes like size, timeouts, and so on
- Pool management operations like refreshing the pool, recycling the pool, and so on
- Monitoring pool statistics and life cycle states