How to handle Timeout due to connection pooling

Last updated by | Subbu Kandhaswamy | Oct 11, 2022 at 10:14 PM PDT

Contents

Connection Pooling

Connection pooling is process of taking a connection from a pool of connections, once connection is created in the pool, then the application can re-use the connection. The very purpose of this is to provide fast & efficient solution, mainly the perform of the system/application depends on the database activity. Making a connection to the SQL database can be time consuming (depending on Network and other latency), and when pooling is enabled (set to true) the request for database connection can be fulfilled from the pool instead of re-connecting to the database, which will increase database performance.

Typical error when connections in the pool finishes

The connections are released back into the pool when you call close or dispose, when no connection available to serve the request will be queued & connection is available in the pool it will be serve to the request. When you do not close connections properly in your application you will get following error:

```
Exception: System.InvalidOperationException
Message: Timeout expired. The timeout period elapsed prior to obtaining a connection from the pool. This may h
```

What controls connection pooling behavior

- Connect Timeout- controls the wait period in seconds when a new connection is requested, if this timeout expires, an exception will be thrown. Default is 15 seconds.
- Max Pool Size- specifies the maximum size of your connection pool. Default is 100.
- Min Pool Size- initial number of connections that will be added to the pool upon its creation. Default is zero; however, you may chose to set this to a small number such as 5 if your application needs consistent response times even after it was idle for hours. In this case the first user requests won't have to wait for those database connections to establish.

- Pooling- controls if your connection pooling on or off. Default as you may've guessed is true. Read on to see when you may use Pooling=false setting.

Sample config - C#

```
Source: System.Data
at  System.Data.SqlClient.SqlConnectionPoolManager.GetPooledConnection(SqlConnectionString options, Boolean& i
at System.Data.SqlClient.SqlConnection.Open()

conn.ConnectionString = "integrated security=SSPI;SERVER=YOUR_SERVER;DATABASE=YOUR_DB_NAME;Min Pool Size=5;Max
```

Notice in this ConnectionString we can set minimum & maximum pool size and also within what time your connection should be time out.

Handling

```
SqlConnection SqlCon = new SqlConnection(myConnectionString);
try
{
          SqlCon.Open();
          DbActivity(SqlCon);
}
finally
{
          SqlCon.Close();
}
```

How to use in (C#) application

```
SqlConnection SqlCon = new SqlConnection();
try
{
SqlCon.ConnectionString = "integrated security=SSPI;SERVER=YOUR_SERVER; DATABASE=YOUR_DB_NAME;Min Pool Size=5;
SqlCon.Open();
}
catch(Exception)
{

}
finally
{
    SqlCon.Close();
}
```

Best practices

- Open a connection only when you need it, not before.
- Close your connection as soon as you are done using it.
- Be sure to close any user-defined transactions before closing a connection.
- Do not close all your connections in the pool, keep at least one connection alive

Reference

- Connection Pooling

How good have you found this content?