DataSource Service

Summary

This service provides connection to database. It supports various types of database connection and abstraction class for this, excluding dependence between task logic and database connection type.

Description

DataSourceimplementions per Connection Provider

This usesDataSource implements with a logic for obtaining Connection object per Connection Provider.

JDBCDataSource

This creates Database Connection using JDBC driver.

Configuration

```
<bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="${driver}" />
        <property name="url" value="${dburl}" />
        <property name="username" value="${username}" />
        <property name="password" value="${password}" />
</bean>
```

PROPERTIES	Description
driverClassName	JDBC driver class name setting
url	JDBC URL to access DataBase
username	User name to access DataBase
password	Password to access DataBase

Sample Source

```
@Resource(name = "dataSource")
DataSourcedataSource;

@Resource(name = "jdbcProperties")
Propertiesjdbc Properties;

booleanisHsql = true;

@Test
public void testJdbcDataSource() throws Exception {

assertNotNull(dataSource);
assertEquals("org.springframework.jdbc.datasource.DriverManagerDataSource",
dataSource.getClass().getName());

Connection con = null;
Statementstmt = null;
ResultSetrs = null;

try {
con = dataSource.getConnection();
assertNotNull(con);
stmt = con.createStatement();
rs = stmt.executeQuery("select 'x' as x from dual");
while (rs.next()) {
assertEquals("x", rs.getString(1));
```

```
                }
               ........
            }
}
```

DBCPDataSource

This is a database connection implement using JDBC driver, as well as a Database Connection Pool of Jakarta called Commons DBCP.

Configuratioin

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="${driver}"/>
        <property name="url" value="${dburl}"/>
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
        <property name="defaultAutoCommit" value="false"/>
        <property name="poolPreparedStatements" value="true"/>
</bean>
```

PROPERTIES	Description
driverClassName	class name setting of jdbc driver
url	Setting DataBaseurl
username	User name to access DataBase
password	Password to access DataBase
defaultAutoCommit	Setting auto-commit or not for the connection returned from datasource
poolPreparedStatements	Whether to use PreparedStatement
maxActive	Setting the maximum number of active connection that can be allocated at the same time
maxIdle	Setting the maximum number of idle connection that can be left at the pool
maxWait	Setting the maximum waiting time if allconnections are used
defaultReadOnly	Granting the read-only property to the connection created by Connection Pool
defaultTransactionIsolation	Granting transaction isolation property for connection returned
defaultCatalog	Setting the catalog of connection
minIdle	Setting the minimum number of idle connection of Connection pool
initialSize	Setting initial connection size to be created at Connection pool
testOnBorrow	Determination of whether to check the validation of the object before getting the object from Connection pool
testOnReturn	Determination of whether to check the validation of object before returning the object
validationQuery	Setting validationQuery
loginTimeout	Setting the login timeout(in seconds) for connection to database

Sample Source

```
@Resource(name = "dataSource")
DataSourcedataSource;

@Resource(name = "jdbcProperties")
PropertiesjdbcProperties;

booleanisHsql = true;

@Test
public void testDbcpDataSource() throws Exception
```

```
{

assertNotNull(dataSource);
assertEquals("org.apache.commons.dbcp.BasicDataSource", dataSource.getClass().getName());

Connection con = null;
Statementstmt = null;
ResultSetrs = null;

try
  {
con = dataSource.getConnection();
assertNotNull(con);
stmt = con.createStatement();
rs = stmt.executeQuery("select 'x' as x from dual");
while (rs.next()) {
assertEquals("x", rs.getString(1));
    }
   } catch (Exception e) {
fail("JdbcDataSource Test Failed! : " + e.getMessage());
e.printStackTrace();
   }
   ........
}
```

C3P0DataSource

This is an implement that creates DataBase Connection using JDBC driver. Matters related to the C3P0 Library can be checked at C3P0 Configuration.

Configuration

```
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
           destroy-method="close">
           <property name="driverClass" value="${driver}" />
           <property name="jdbcUrl" value="${dburl}" />
           <property name="user" value="${username}" />
           <property name="password" value="${password}" />
           <property name="initialPoolSize" value="3" />
           <property name="minPoolSize" value="3" />
           <property name="maxPoolSize" value="50" />
           <!--<property name="timeout" value="0" /> --><!-- 0 means: no timeout -->
           <property name="idleConnectionTestPeriod" value="200" />
           <property name="acquireIncrement" value="1" />
           <property name="maxStatements" value="0" /><!-- 0 means: statement caching is turned
off.  -->
           <!-- c3p0 is very asynchronous. Slow JDBC operations are generally performed
by helper threads that don't hold contended locks.
                      Spreading these operations over multiple threads can significantly improve
performance
                      by allowing multiple operations to be performed simultaneously -->
           <property name="numHelperThreads" value="3" /><!-- 3 is default -->
</bean>
```

PROPERTIES	Description
driverClass	jdbc driver
jdbcUrl	DB URL
user	User name
password	Password
initialPoolSize	Pool initial value
minPoolSize	Pool minimum value

maxPoolSize	Pool maximum value
idleConnectionTestPeriod	Idle status test time
acquireIncrement	Increment
maxStatements	Whether to keep cache or not
numHelperThreads	Number of HelperThread

Sample Source

```
@Resource(name = "dataSource")
DataSourcedataSource;

@Resource(name = "jdbcProperties")
PropertiesjdbcProperties;

@Test
public void testC3p0DataSource() throws Exception
{

assertNotNull(dataSource);
assertEquals("com.mchange.v2.c3p0.ComboPooledDataSource", dataSource.getClass().getName());

Connection con = null;
Statementstmt = null;
ResultSetrs = null;

try {
con = dataSource.getConnection();
assertNotNull(con);
stmt = con.createStatement();
rs = stmt.executeQuery("select 'x' as x from dual");
while (rs.next()) {
assertEquals("x", rs.getString(1));
            }
        } catch (Exception e) {
fail("JdbcDataSource Test Failed! : " + e.getMessage());
e.printStackTrace();
        }
  ..................
}
```

JNDIDataSource

This creates Database Connection using JNDI Lookup. JNDIDataSource imports DataSource from JNDI tree provided by Enterprise application server.

Configuration

Jeus Setting

```
<jee:jndi-lookup id="dataSource" jndi-name="${jndiName}" resource-ref="true">
<jee:environment>
          java.naming.factory.initial=${jeus.java.naming.factory.initial}
            java.naming.provider.url=${jeus.java.naming.provider.url}
</jee:environment>
</jee:jndi-lookup>
```

Weblogic Setting

```
<util:properties id="jndiProperties" location="classpath:/META-INF/spring/jndi.properties" />
```

```
<jee:jndi-lookup id="dataSource" jndi-name="${jndiName}" resource-ref="true" environment-ref="jndiProperties" />
```

PROPERTIES	**Description**
jndiTemplate	Setting JNDI template for JNDI setting
jndiEnvironment	Setting JNDI environment for searching JNDI
resourceRef	Setting whether J2EE can be searched in container
expectedType	Designation of the type of JNDI object
jndiName	Setting JNDI name for searching
proxyInterface	Setting the proxy interface for using JNDI object
lookupOnStartup	Setting whether to search JNDI object at starup
cache	Whether to cache JNDI objects
defaultObject	Designation of the default object to deliver when JNDI lookup fails

Sample Source

```
@Resource(name = "dataSource")
DataSourcedataSource;

@Resource(name = "jdbcProperties")
Propertiesjdbcproperties;

@Test
public void testJndiJeusDataSource() throws Exception
{

assertNotNull(dataSource);
assertEquals("jeus.jdbc.connectionpool.DataSourceWrapper", dataSource.getClass().getName());

Connection con = null;
Statementstmt = null;
ResultSetrs = null;

try {
con = dataSource.getConnection();
assertNotNull(con);
stmt = con.createStatement();
rs = stmt.executeQuery("select 'x' as x from dual");
while (rs.next()) {
assertEquals("x", rs.getString(1));
            }
        } catch (Exception e) {
fail("JdbcDataSource Test Failed! : " + e.getMessage());
e.printStackTrace();
        }
        ..................
}
```

- Jeus5.0 datasource : jeus.jdbc.driver.oracle.OracleConnectionPool
- Jeus6.0 datasource : jeus.jdbc.connectionpool.DataSourceWrapper

```
@Resource(name = "dataSource")
DataSourcedataSource;

@Resource(name = "jdbcProperties")
Propertiesjdbcproperties;

@Test
public void testJndiDataSource() throws Exception
{
```

```
assertNotNull(dataSource);
assertEquals("weblogic.jdbc.common.internal.RmiDataSource_922_WLStub",
dataSource.getClass().getName());

Connection con = null;
Statementstmt = null;
ResultSetrs = null;

try {
con = dataSource.getConnection();
assertNotNull(con);
stmt = con.createStatement();
rs = stmt.executeQuery("select 'x' as x from dual");
while (rs.next()) {
assertEquals("x", rs.getString(1));
               }
          } catch (Exception e) {
fail("JdbcDataSource Test Failed! : " + e.getMessage());
e.printStackTrace();
          }
          ..................
}
```

- Weblogicdatasource

Reference