org.springframework.jdbc.core

Interface StatementCallback<T>

```
public interface StatementCallback<T>
```

Generic callback interface for code that operates on a JDBC Statement. Allows to execute any number of operations on a single Statement, for example a single `executeUpdate` call or repeated `executeUpdate` calls with varying SQL.

Used internally by JdbcTemplate, but also useful for application code.

Since:

16.03.2004

Author:

Juergen Hoeller

See Also:

`JdbcTemplate.execute(StatementCallback)`

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method and Description
T	`doInStatement(Statement stmt)` Gets called by `JdbcTemplate.execute` with an active JDBC Statement.

Method Detail

doInStatement

```
T doInStatement(Statement stmt)
         throws SQLException,
                DataAccessException
```

Gets called by `JdbcTemplate.execute` with an active JDBC Statement. Does not need to care about closing the Statement or the Connection, or about handling transactions: this will all be handled by Spring's JdbcTemplate.

NOTE: Any ResultSets opened should be closed in finally blocks within the callback implementation. Spring will close the Statement object after the callback returned, but this does not necessarily imply that the ResultSet resources will be closed: the Statement objects might get pooled by the connection pool, with `close` calls only returning the object to the pool but not physically closing the resources.

If called without a thread-bound JDBC transaction (initiated by DataSourceTransactionManager), the code will simply get executed on the JDBC connection with its transactional semantics. If JdbcTemplate is configured to use a JTA-aware DataSource, the JDBC connection and thus the callback code will be transactional if a JTA transaction is active.

Allows for returning a result object created within the callback, i.e. a domain object or a collection of domain objects. Note that there's special support for single step actions: see JdbcTemplate.queryForObject etc. A thrown RuntimeException is treated as application exception, it gets propagated to the caller of the template.

Parameters:

`stmt - active JDBC Statement`

Returns:

`a result object, or null if none`

Throws:

`SQLException - if thrown by a JDBC method, to be auto-converted to a DataAccessException by a SQLExceptionTranslator`

`DataAccessException - in case of custom exceptions`

See Also:

`JdbcTemplate.queryForObject(String, Class), JdbcTemplate.queryForRowSet(String)`