UpdateableDataContext

The `UpdateableDataContext` interface is a sub-interface of DataContext, and represents any DataContext that is capable of making changes/updates to the data.

The single method for doing updates is executeUpdate(UpdateScript). The UpdateScript argument describes the sequence of updates that you want to perform.

Single updates

There are a number of built-in implementations of UpdateScript which provide out-of-the-box single/atomic updates:

```
UpdateableDataContext dataContext = ...;

// InsertInto update:
// INSERT INTO table (name,age) VALUES ('Polly the Sheep', -1)
dataContext.executeUpdate(new InsertInto(table).value("name", "Polly the Sheep").value("age", -1));

// Update update:
// UPDATE table SET age = 10 WHERE name = 'Polly the Sheep'
dataContext.executeUpdate(new Update(table).where("name").eq("Polly the Sheep").value("age", 10));

// DeleteFrom update:
// DELETE FROM table WHERE name = 'Polly the Sheep'
dataContext.executeUpdate(new DeleteFrom(table).where("name").eq("Polly the Sheep"));
```

Custom update script

You can also bundle together updates into a single update script. This will typically mean that transactional attributes are improved and resources remain open during the update, a bit depending on the datastore / DataContext implementation.

```
UpdateableDataContext dataContext = ...;
dataContext.executeUpdate(new UpdateScript() {
        public void run(UpdateCallback callback) {
                // INSERT INTO table (name,age) VALUES ('Polly the Sheep', -1)
                callback.insertInto(table).value("name", "Polly the Sheep").value("age", -1)).execute();

                // UPDATE table SET age = 10 WHERE name = 'Polly the Sheep'
                callback.update(table).where("name").eq("Polly the Sheep").value("age", 10)).execute();

                // DELETE FROM table WHERE name = 'Polly the Sheep'
                callback.deleteFrom(table).where("name").eq("Polly the Sheep")).execute();
        };
});
```

Notice also that the UpdateScript interface is a single-method-interface (SAM). This means that from Java 8 and forward, you should be able to provide a closure here instead of an anonymous inner class.

See also

You may be interested in:

- Schema and table semantics - to see what it means to "create a table" in different implementations.