Compiled queries

MetaModel supports 'compiling' queries which means preparing them for reuse multiple times, ahead of execution time. This mechanism resembles what is in JDBC terms 'Prepared Statements', and is in deed implemented this way for the JDBC adapter.

Compiled queries are abstract, meaning they can have parameters which are provided at execution time. The advantage of this is that the same query can be reused for many (but similar) purposes. If your application needs to perform a lookup of records with slightly varying features (eg. different keys or different attributes), a compiled query can greatly improve performance, since many database backends can optimize the server-side query planning.

Compiled queries have life-cycles that can span small jobs, large batch jobs or entire application lifetimes. We recommend treating them like other compileable resources, eg. regular expressions or input streams - create them when needed for a particular job, use and reuse them for the duration of this job, and then close them. This will ensure that your application has an elastic behaviour and that it does not needlessly take up resources on the (database) server for longer than needed.

Below is an example of compiling a query:

```
DataContext dataContext = ...
Query query = dataContext.query().from("persons").select("name").where("id").eq(new QueryParameter()).toQuery();
CompiledQuery compiledQuery = dataContext.compileQuery(query);
```

To use the compiled query, fire it with the `executeQuery` along with a parameter list that matches the `QueryParameters` in the original query object.

Here's an example of (re)using the compiled query with 3 different parameter values. Notice also the try-finally construct which ensures closing the compiled queries after use. Obviously, you can also choose to keep the compiled query alive for longer than this, but closing it does free up resources on both client and server, so consider whether it is feasible to close it or not on a more regular basis.

```
try {
   DataSet dataSet1 = dataContext.executeQuery(compiledQuery, 123);
   DataSet dataSet2 = dataContext.executeQuery(compiledQuery, 124);
   DataSet dataSet3 = dataContext.executeQuery(compiledQuery, 125);
} finally {
   compiledQuery.close();
}
```