DataContext

The `DataContext` interface is probably the most central construct in MetaModel. The `DataContext` can be compared to a "connection" to the datastore that you're working with. The `DataContext` exposes methods to explore the schema model and to query the data of a datastore.

The sub-type UpdateableDataContext further adds the ability to update/change data contained in the datastore.

Exposing the schema model (read more)

Methods:

- `getSchemas(): Schema[]`
- `getSchemaNames(): String[]`
- `getDefaultSchema(): Schema`
- `getSchemaByName(String): Schema`
- `getTableByQualifiedLabel(String): Table`
- `getColumnByQualifiedLabel(String): Column`

These methods allow you to explore the schema model of your datastore. You can either traverse with e.g. *getSchemas* (and in turn use the traversal methods of Schema and Table classes) or find specific elements using e.g. *getSchemaByName*, *getTableByQualifiedLabel* and *getColumnByQualifiedLabel*.

Querying for data (read more)

Methods:

- `parseQuery(String): Query`
- `executeQuery(Query): DataSet`
- `executeQuery(String): DataSet`
- `query(): (fluent query builder return-types)`

When we say that MetaModel offers a type-safe querying API, we mean that you can use the traversed schema model to build your queries. That way you can build queries that refer columns and tables that are guaranteed to exist and be correctly referred, since they where provided according to the metadata of the datastore.