S.O.D.A. Simple Object Database Access - design documentation v. (unknown)

September 20, 2002

Contents

Class Hierarchy

Interfaces

- java.io.Serializable
    - org.odbms.Evaluation (in 1.1.4, page 13)
- org.odbms.Candidate (in 1.1.1, page 8)
- org.odbms.Constraint (in 1.1.2, page 9)
    - org.odbms.Constraints (in 1.1.3, page 12)
- org.odbms.ObjectContainer (in 1.1.5, page 14)
- org.odbms.ObjectSet (in 1.1.6, page 15)
- org.odbms.Query (in 1.1.7, page 16)

Chapter 1

Package org.odbms

Package Contents *Page*

1.1 Interfaces

1.1.1 *Interface* Candidate

candidate for Evaluation (in 1.1.4, page 13)callbacks.

During query execution (in 1.1.7, page 17)all registered Evaluation (in 1.1.4, page 13)callback handlers are called with Candidate (in 1.1.1, page 8)proxies that represent the persistent objects that meet all other Query (in 1.1.7, page 16)criteria.

A Candidate (in 1.1.1, page 8)provides access to the persistent object it represents and allows to specify, whether it is to be included in the ObjectSet (in 1.1.6, page 15)resultset.

Declaration

```
public interface Candidate
```

Method summary

getObject() returns the persistent object that is represented by this query Candidate (in 1.1.1, page 8).
include(boolean) specify whether the Candidate is to be included in the ObjectSet (in 1.1.6, page 15)resultset.

Methods

- *getObject*
  `java.lang.Object` **`getObject( )`**
  - **Description**
    returns the persistent object that is represented by this query Candidate (in 1.1.1, page 8).
  - **Returns** – Object the persistent object.

- *include*
  `void` **include**`( boolean` **flag** `)`
  - **Description**
    specify whether the Candidate is to be included in the ObjectSet (in 1.1.6, page 15)resultset.

    This method may be called multiple times. The last call prevails.
  - **Parameters**
    * `flag` – inclusion.

1.1.2 *Interface* Constraint

constraint to limit the objects returned upon query execution (in 1.1.7, page 17).

Constraints are constructed by calling Query.constrain() (in 1.1.7, page 16).

Constraints can be joined with the methods and() (in 1.1.2, page 9)and or() (in 1.1.2, page 11).

The methods to modify the constraint evaluation algorithm may be merged, to construct combined evaluation rules. Examples:

- `Constraint#smaller().equal()` for "smaller or equal"
- `Constraint#not().like()` for "not like"
- `Constraint#not().greater().equal()` for "not greater or equal"

Declaration

```
public interface Constraint
```

All known subclasses

Constraints (in 1.1.3, page 12)

All known subinterfaces

Constraints (in 1.1.3, page 12)

Method summary

and(Constraint) links two Constraints for AND evaluation.
contains() sets the evaluation mode to containment comparison.
equal() sets the evaluation mode to ==.
getObject() returns the Object the query graph was constrained with to create this Constraint (in 1.1.2, page 9).
greater() sets the evaluation mode to >.
identity() sets the evaluation mode to identity comparison.
like() sets the evaluation mode to "like" comparison.
not() turns on not() comparison.
or(Constraint) links two Constraints for OR evaluation.
smaller() sets the evaluation mode to <.

Methods

- *and*
  `Constraint and( Constraint with )`
  - **Description**
    links two Constraints for AND evaluation.
  - **Parameters**

    * `with` – the other Constraint (in 1.1.2, page 9)
  - **Returns** – a new Constraint (in 1.1.2, page 9), that can be used for further calls to and() (in 1.1.2, page 9)and or() (in 1.1.2, page 11)

- *contains*
  `Constraint` **contains**`( )`

  - **Description**
    sets the evaluation mode to containment comparison.

    Evaluation is dependant on the constrained query node:

`String`	the persistent object is tested to contain a substring.
ava.util.Collection collections	the persistent object is tested to contain all elements of the constraining object.

  - **Returns** – this Constraint (in 1.1.2, page 9)to allow the chaining of method calls.

- *equal*
  `Constraint` **equal**`( )`

  - **Description**
    sets the evaluation mode to `==`.
  - **Returns** – this Constraint (in 1.1.2, page 9)to allow the chaining of method calls.

- *getObject*
  `java.lang.Object` **getObject**`( )`

  - **Description**
    returns the Object the query graph was constrained with to create this Constraint (in 1.1.2, page 9).
  - **Returns** – Object the constraining object.

- *greater*
  `Constraint` **greater**`( )`

  - **Description**
    sets the evaluation mode to >.
  - **Returns** – this Constraint (in 1.1.2, page 9)to allow the chaining of method calls.

- *identity*
  `Constraint` **identity**`( )`

  - **Description**
    sets the evaluation mode to identity comparison.
  - **Returns** – this Constraint (in 1.1.2, page 9)to allow the chaining of method calls.

- *like*
  `Constraint` **like**`( )`

  - **Description**
    sets the evaluation mode to ”like” comparison.
  - **Returns** – this Constraint (in 1.1.2, page 9)to allow the chaining of method calls.

- *not*
  `Constraint not( )`
  - **Description**
    turns on not() comparison.
  - **Returns** – this Constraint (in 1.1.2, page 9)to allow the chaining of method calls.

- *or*
  `Constraint or( Constraint with )`
  - **Description**
    links two Constraints for OR evaluation.
  - **Parameters**
    * `with` – the other Constraint (in 1.1.2, page 9)
  - **Returns** – a new Constraint (in 1.1.2, page 9), that can be used for further calls to and() (in 1.1.2, page 9)and or() (in 1.1.2, page 11)

- *smaller*
  `Constraint smaller( )`
  - **Description**
    sets the evaluation mode to <.
  - **Returns** – this Constraint (in 1.1.2, page 9)to allow the chaining of method calls.

1.1.3 *Interface* **Constraints**

set of Constraint (in 1.1.2, page 9)objects.

This extension of the Constraint (in 1.1.2, page 9)interface allows setting the evaluation mode of all contained Constraint (in 1.1.2, page 9)objects with single calls.

See also Query#constraints() (in 1.1.7, page 17).

Declaration

public interface Constraints
implements Constraint

Method summary

toArray() returns an array of the contained Constraint (in 1.1.2, page 9)objects.

Methods

- *toArray*
  `Constraint[] toArray( )`
  - **Description**
    returns an array of the contained Constraint (in 1.1.2, page 9)objects.
  - **Returns** – an array of the contained Constraint (in 1.1.2, page 9)objects.

1.1.4 *Interface* **Evaluation**

for implementation of callback evaluations.

To constrain a Query (in 1.1.7, page 16)node with your own callback `Evaluation`, construct an object that implements the `Evaluation` interface and register it by passing it to Query#constrain(Object) (in 1.1.7, page 16).

Evaluations are called as the last step during query execution, after all other constraints have been applied. Evaluations in higher level Query (in 1.1.7, page 16)nodes in the query graph are called first.

Declaration

public interface Evaluation
implements java.io.Serializable

Method summary

evaluate(Candidate) callback method during query execution (in 1.1.7, page 17).

Methods

- *evaluate*
  `void` **evaluate(** `Candidate` **candidate** `)`
  - **Description**
    callback method during query execution (in 1.1.7, page 17).
  - **Parameters**
    * `Candidate` – reference to the candidate persistent object.

1.1.5 *Interface* **ObjectContainer**

database engine interface.

The `ObjectContainer` interface provides all methods to store, retrieve and delete objects and to change object state.

Declaration

```
public interface ObjectContainer
```

Method summary

query() factory method to create a new `Query` (at Query.html) object to query this ObjectContainer.

Methods

- *query*
  `Query` **query**`( )`
  - **Description**
    factory method to create a new `Query` (at Query.html) object to query this ObjectContainer.
  - **Returns** – a new Query object

1.1.6 *Interface* **ObjectSet**

query resultset.

The `ObjectSet` interface serves as a cursor to iterate through a set of objects retrieved by a query.

Declaration

```
public interface ObjectSet
```

Method summary

hasNext() returns `true` if the `ObjectSet` has more elements.
next() returns the next object in the `ObjectSet`.
reset() resets the `ObjectSet` cursor before the first element.
size() returns the number of elements in the `ObjectSet`.

Methods

- *hasNext*
  `boolean` **hasNext**`( )`
  - **Description**
    returns `true` if the `ObjectSet` has more elements.
  - **Returns** – boolean `true` if the `ObjectSet` has more elements.

- *next*
  `java.lang.Object` **next**`( )`
  - **Description**
    returns the next object in the `ObjectSet`.
  - **Returns** – the next object in the `ObjectSet`.

- *reset*
  `void` **reset**`( )`
  - **Description**
    resets the `ObjectSet` cursor before the first element.

    A subsequent call to `next()` will return the first element.

- *size*
  `int` **size**`( )`
  - **Description**
    returns the number of elements in the `ObjectSet`.
  - **Returns** – the number of elements in the `ObjectSet`.

1.1.7 *Interface* **Query**

handle to a node in the query graph.

A node in the query graph can represent multiple classes, one class or an attribute of a class.

The graph is automatically extended with attributes of added constraints (see constrain() (in 1.1.7, page 16)) and upon calls to descend() (in 1.1.7, page 17)that request nodes that do not yet exist.

References to joined nodes in the query graph kann be obtained by "walking" along the nodes of the graph with the method descend() (in 1.1.7, page 17).

#execute() (in 1.1.7, page 17)evaluates the entire graph against all persistent objects.

#execute() (in 1.1.7, page 17)can be called from any Query (in 1.1.7, page 16)node of the graph. It will return an ObjectSet (in 1.1.6, page 15)filled with objects of the class/classes that the node, it was called from, represents.

Declaration

```
public interface Query
```

Method summary

constrain(Object) adds a constraint to this node.
constraints() returns a Constraints (in 1.1.3, page 12)object that holds an array of all constraints on this node.
descend(String) returns a reference to a descendant node in the query graph.
execute() executes the Query (in 1.1.7, page 16).
orderAscending() adds an ascending ordering criteria to this node of the query graph.
orderDescending() adds a descending order criteria to this node of the query graph.

Methods

- *constrain*
  `Constraint` **`constrain(`** `java.lang.Object` **`constraint`** `)`
  - **Description**
    adds a constraint to this node.

    If the constraint contains attributes that are not yet present in the query graph, the query graph is extended accordingly.

    Special behaviour for:
    * class Class : confine the result to objects of one class (if the Class object represents a class) or to objects implementing a specific interface (if the Class object represents an interface).
    * interface Evaluation (in 1.1.4, page 13): run evaluation callbacks against all candidates.

- **Parameters**
  - `constraint` – the constraint to be added to this Query.
- **Returns** – a new Constraint (in 1.1.2, page 9)for this query node or `null` for objects implementing the Evaluation (in 1.1.4, page 13)interface.

- *constraints*
  `Constraints` **constraints**`( )`
  - **Description**
    returns a Constraints (in 1.1.3, page 12)object that holds an array of all constraints on this node.
  - **Returns** – on this query node.

- *descend*
  `Query` **descend**`( java.lang.String` **fieldName** `)`
  - **Description**
    returns a reference to a descendant node in the query graph.

    If the node does not exist, it will be created.

    All classes represented in the query node are tested, whether they contain a field with the specified field name. The descendant Query node will be created from all possible candidate classes.
  - **Parameters**
    - `field` – path to the descendant.
  - **Returns** – descendant Query (in 1.1.7, page 16)node

- *execute*
  `ObjectSet` **execute**`( )`
  - **Description**
    executes the Query (in 1.1.7, page 16).
  - **Returns** – - the result of the Query (in 1.1.7, page 16).

- *orderAscending*
  `Query` **orderAscending**`( )`
  - **Description**
    adds an ascending ordering criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called.
  - **Returns** – this Query (in 1.1.7, page 16)object to allow the chaining of method calls.

- *orderDescending*
  `Query` **orderDescending**`( )`
  - **Description**
    adds a descending order criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called.
  - **Returns** – this Query (in 1.1.7, page 16)object to allow the chaining of method calls.

Index