javax.persistence

Interface StoredProcedureQuery

All Superinterfaces:

Query

```
public interface StoredProcedureQuery
extends Query
```

Interface used to control stored procedure query execution.

Stored procedure query execution may be controlled in accordance with the following:

- The `setParameter` methods are used to set the values of all required `IN` and `INOUT` parameters. It is not required to set the values of stored procedure parameters for which default values have been defined by the stored procedure.
- When `getResultList` and `getSingleResult` are called on a `StoredProcedureQuery` object, the provider will call `execute` on an unexecuted stored procedure query before processing `getResultList` or `getSingleResult`.
- When `executeUpdate` is called on a `StoredProcedureQuery` object, the provider will call `execute` on an unexecuted stored procedure query followed by `getUpdateCount`. The results of `executeUpdate` will be those of `getUpdateCount`.
- The `execute` method supports both the simple case where scalar results are passed back only via `INOUT` and `OUT` parameters as well as the most general case (multiple result sets and/or update counts, possibly also in combination with output parameter values).
- The `execute` method returns true if the first result is a result set, and false if it is an update count or there are no results other than through `INOUT` and `OUT` parameters, if any.
- If the `execute` method returns true, the pending result set can be obtained by calling `getResultList` or `getSingleResult`.
- The `hasMoreResults` method can then be used to test for further results.
- If `execute` or `hasMoreResults` returns false, the `getUpdateCount` method can be called to obtain the pending result if it is an update count. The `getUpdateCount` method will return either the update count (zero or greater) or -1 if there is no update count (i.e., either the next result is a result set or there is no next update count).
- For portability, results that correspond to JDBC result sets and update counts need to be processed before the values of any `INOUT` or `OUT` parameters are extracted.
- After results returned through `getResultList` and `getUpdateCount` have been exhausted, results returned through `INOUT` and `OUT` parameters can be retrieved.
- The `getOutputParameterValue` methods are used to retrieve the values passed back from the procedure through `INOUT` and `OUT` parameters.
- When using `REF_CURSOR` parameters for result sets the update counts should be exhausted before calling `getResultList` to retrieve the result set. Alternatively, the `REF_CURSOR` result set can be retrieved through `getOutputParameterValue`. Result set mappings will be applied to results corresponding to `REF_CURSOR` parameters in the order the parameters were registered with the query.
- In the simplest case, where results are returned only via `INOUT` and `OUT` parameters, `execute` can be followed immediately by calls to `getOutputParameterValue`.

Since:

Java Persistence 2.1

See Also:

`Query`, `Parameter`

Method Summary

Methods

Modifier and Type	Method and Description
boolean	**execute**() Return true if the first result corresponds to a result set, and false if it is an update count or if there are no results.
int	**executeUpdate**() Return the update count of -1 if there is no pending result or if the first result is not an update count.
java.lang.Object	**getOutputParameterValue**(int position) Retrieve a value passed back from the procedure through an INOUT or OUT parameter.
java.lang.Object	**getOutputParameterValue**(java.lang.String parameterName) Retrieve a value passed back from the procedure through an INOUT or OUT parameter.
java.util.List	**getResultList**() Retrieve the list of results from the next result set.
java.lang.Object	**getSingleResult**() Retrieve a single result from the next result set.
int	**getUpdateCount**() Return the update count or -1 if there is no pending result or if the next result is not an update count.
boolean	**hasMoreResults**() Return true if the next result corresponds to a result set, and false if it is an update count or if there are no results other than through INOUT and OUT parameters, if any.
StoredProcedureQuery	**registerStoredProcedureParameter**(int position, java.lang.Class type, **ParameterMode** mode) Register a positional parameter.
StoredProcedureQuery	**registerStoredProcedureParameter**(java.lang.String parameterName, java.lang.Class type, **ParameterMode** mode) Register a named parameter.
StoredProcedureQuery	**setFlushMode**(**FlushModeType** flushMode) Set the flush mode type to be used for the query execution.
StoredProcedureQuery	**setHint**(java.lang.String hintName, java.lang.Object value) Set a query property or hint.
StoredProcedureQuery	**setParameter**(int position, java.util.Calendar value, **TemporalType** temporalType) Bind an instance of java.util.Calendar to a positional parameter.
StoredProcedureQuery	**setParameter**(int position, java.util.Date value, **TemporalType** temporalType) Bind an instance of java.util.Date to a positional parameter.
StoredProcedureQuery	**setParameter**(int position, java.lang.Object value) Bind an argument value to a positional parameter.
StoredProcedureQuery	**setParameter**(**Parameter**<java.util.Calendar> param, java.util.Calendar value, **TemporalType** temporalType) Bind an instance of java.util.Calendar to a Parameter object.
StoredProcedureQuery	**setParameter**(**Parameter**<java.util.Date> param, java.util.Date value, **TemporalType** temporalType) Bind an instance of java.util.Date to a Parameter object.
<T> **StoredProcedureQuery**	**setParameter**(**Parameter**<T> param, T value) Bind the value of a Parameter object.
StoredProcedureQuery	**setParameter**(java.lang.String name, java.util.Calendar value, **TemporalType** temporalType) Bind an instance of java.util.Calendar to a named parameter.

StoredProcedureQuery	**setParameter**(java.lang.String name, java.util.Date value, **TemporalType** temporalType) Bind an instance of java.util.Date to a named parameter.
StoredProcedureQuery	**setParameter**(java.lang.String name, java.lang.Object value) Bind an argument value to a named parameter.

Methods inherited from interface javax.persistence.Query

getFirstResult, getFlushMode, getHints, getLockMode, getMaxResults, getParameter, getParameter, getParameter, getParameter, getParameters, getParameterValue, getParameterValue, getParameterValue, isBound, setFirstResult, setLockMode, setMaxResults, unwrap

Method Detail

setHint

```
StoredProcedureQuery setHint(java.lang.String hintName,
                             java.lang.Object value)
```

Set a query property or hint. The hints elements may be used to specify query properties and hints. Properties defined by this specification must be observed by the provider. Vendor-specific hints that are not recognized by a provider must be silently ignored. Portable applications should not rely on the standard timeout hint. Depending on the database in use, this hint may or may not be observed.

Specified by:

setHint in interface Query

Parameters:

hintName - name of the property or hint

value - value for the property or hint

Returns:

the same query instance

Throws:

java.lang.IllegalArgumentException - if the second argument is not valid for the implementation

setParameter

```
<T> StoredProcedureQuery setParameter(Parameter<T> param,
                                      T value)
```

Bind the value of a Parameter object.

Specified by:

setParameter in interface Query

Parameters:

param - parameter object

value - parameter value

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if the parameter does not correspond to a parameter of the query

setParameter

```
StoredProcedureQuery setParameter(Parameter<java.util.Calendar> param,
                                  java.util.Calendar value,
                                  TemporalType temporalType)
```

Bind an instance of `java.util.Calendar` to a `Parameter` object.

Specified by:

`setParameter` in interface `Query`

Parameters:

`param` - parameter object

`value` - parameter value

`temporalType` - temporal type

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if the parameter does not correspond to a parameter of the query

setParameter

```
StoredProcedureQuery setParameter(Parameter<java.util.Date> param,
                                  java.util.Date value,
                                  TemporalType temporalType)
```

Bind an instance of `java.util.Date` to a `Parameter` object.

Specified by:

`setParameter` in interface `Query`

Parameters:

`param` - parameter object

`value` - parameter value

`temporalType` - temporal type

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if the parameter does not correspond to a parameter of the query

setParameter

```
StoredProcedureQuery setParameter(java.lang.String name,
                                  java.lang.Object value)
```

Bind an argument value to a named parameter.

Specified by:

`setParameter` in interface `Query`

Parameters:

`name` - parameter name

`value` - parameter value

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if the parameter name does not correspond to a parameter of the query or if the argument is of incorrect type

setParameter

```
StoredProcedureQuery setParameter(java.lang.String name,
                                  java.util.Calendar value,
                                  TemporalType temporalType)
```

Bind an instance of `java.util.Calendar` to a named parameter.

Specified by:

`setParameter` in interface `Query`

Parameters:

`name` - parameter name

`value` - parameter value

`temporalType` - temporal type

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if the parameter name does not correspond to a parameter of the query or if the value argument is of incorrect type

setParameter

```
StoredProcedureQuery setParameter(java.lang.String name,
                                  java.util.Date value,
                                  TemporalType temporalType)
```

Bind an instance of `java.util.Date` to a named parameter.

Specified by:

`setParameter` in interface `Query`

Parameters:

`name` - parameter name

`value` - parameter value

`temporalType` - temporal type

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if the parameter name does not correspond to a parameter of the query or if the value argument is of incorrect type

setParameter

```
StoredProcedureQuery setParameter(int position,
                                  java.lang.Object value)
```

Bind an argument value to a positional parameter.

Specified by:

`setParameter` in interface `Query`

Parameters:

`position` - position

`value` - parameter value

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if position does not correspond to a positional parameter of the query or if the argument is of incorrect type

setParameter

```
StoredProcedureQuery setParameter(int position,
                                  java.util.Calendar value,
                                  TemporalType temporalType)
```

Bind an instance of `java.util.Calendar` to a positional parameter.

Specified by:

`setParameter` in interface `Query`

Parameters:

`position` - position

`value` - parameter value

`temporalType` - temporal type

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if position does not correspond to a positional parameter of the query or if the value argument is of incorrect type

setParameter

```
StoredProcedureQuery setParameter(int position,
                                  java.util.Date value,
                                  TemporalType temporalType)
```

Bind an instance of `java.util.Date` to a positional parameter.

Specified by:

`setParameter` in interface `Query`

Parameters:

`position` - position

`value` - parameter value

`temporalType` - temporal type

Returns:

the same query instance

Throws:

`java.lang.IllegalArgumentException` - if position does not correspond to a positional parameter of the query or if the value argument is of incorrect type

setFlushMode

```
StoredProcedureQuery setFlushMode(FlushModeType flushMode)
```

Set the flush mode type to be used for the query execution. The flush mode type applies to the query regardless of the flush mode type in use for the entity manager.

Specified by:

`setFlushMode` in interface `Query`

Parameters:

`flushMode` - flush mode

Returns:

the same query instance

registerStoredProcedureParameter

```
StoredProcedureQuery registerStoredProcedureParameter(int position,
                                                      java.lang.Class type,
                                                      ParameterMode mode)
```

Register a positional parameter. All parameters must be registered.

Parameters:

`position` - parameter position

`type` - type of the parameter

`mode` - parameter mode

Returns:

the same query instance

registerStoredProcedureParameter

```
StoredProcedureQuery registerStoredProcedureParameter(java.lang.String parameterName,
                                                      java.lang.Class type,
                                                      ParameterMode mode)
```

Register a named parameter.

Parameters:

`parameterName` - name of the parameter as registered or specified in metadata

`type` - type of the parameter

`mode` - parameter mode

Returns:

the same query instance

getOutputParameterValue

```
java.lang.Object getOutputParameterValue(int position)
```

Retrieve a value passed back from the procedure through an INOUT or OUT parameter. For portability, all results corresponding to result sets and update counts must be retrieved before the values of output parameters.

Parameters:

`position` - parameter position

Returns:

the result that is passed back through the parameter

Throws:

`java.lang.IllegalArgumentException` - if the position does not correspond to a parameter of the query or is not an INOUT or OUT parameter

getOutputParameterValue

```
java.lang.Object getOutputParameterValue(java.lang.String parameterName)
```

Retrieve a value passed back from the procedure through an INOUT or OUT parameter. For portability, all results corresponding to result sets and update counts must be retrieved before the values of output parameters.

Parameters:

`parameterName` - name of the parameter as registered or specified in metadata

Returns:

the result that is passed back through the parameter

Throws:

`java.lang.IllegalArgumentException` - if the parameter name does not correspond to a parameter of the query or is not an INOUT or OUT parameter

execute

```
boolean execute()
```

Return true if the first result corresponds to a result set, and false if it is an update count or if there are no results. other than through INOUT and OUT parameters, if any.

Returns:

true if first result corresponds to result set

Throws:

`QueryTimeoutException` - if the query execution exceeds the query timeout value set and only the statement is rolled back

`PersistenceException` - if the query execution exceeds the query timeout value set and the transaction is rolled back

executeUpdate

```
int executeUpdate()
```

Return the update count of -1 if there is no pending result or if the first result is not an update count. The provider will call `execute` on the query if needed.

Specified by:

`executeUpdate` in interface `Query`

Returns:

the update count or -1 if there is no pending result or if the next result is not an update count.

Throws:

`TransactionRequiredException` - if there is no transaction or the persistence context has not been joined to the transaction

`QueryTimeoutException` - if the statement execution exceeds the query timeout value set and only the statement is rolled back

`PersistenceException` - if the query execution exceeds the query timeout value set and the transaction is rolled back

getResultList

```
java.util.List getResultList()
```

Retrieve the list of results from the next result set. The provider will call `execute` on the query if needed. A `REF_CURSOR` result set, if any, will be retrieved in the order the `REF_CURSOR` was registered with the query.

Specified by:

`getResultList` in interface `Query`

Returns:

a list of the results or null is the next item is not a result set

Throws:

`QueryTimeoutException` - if the query execution exceeds the query timeout value set and only the statement is rolled back

`PersistenceException` - if the query execution exceeds the query timeout value set and the transaction is rolled back

getSingleResult

```
java.lang.Object getSingleResult()
```

Retrieve a single result from the next result set. The provider will call `execute` on the query if needed. A `REF_CURSOR` result set, if any, will be retrieved in the order the `REF_CURSOR` was registered with the query.

Specified by:

`getSingleResult` in interface `Query`

Returns:

the result or null if the next item is not a result set

Throws:

`NoResultException` - if there is no result in the next result set

`NonUniqueResultException` - if more than one result

`QueryTimeoutException` - if the query execution exceeds the query timeout value set and only the statement is rolled back

`PersistenceException` - if the query execution exceeds the query timeout value set and the transaction is rolled back

hasMoreResults

```
boolean hasMoreResults()
```

Return true if the next result corresponds to a result set, and false if it is an update count or if there are no results other than through INOUT and OUT parameters, if any.

Returns:

true if next result corresponds to result set

Throws:

`QueryTimeoutException` - if the query execution exceeds the query timeout value set and only the statement is rolled back

`PersistenceException` - if the query execution exceeds the query timeout value set and the transaction is rolled back

getUpdateCount

```
int getUpdateCount()
```

Return the update count or -1 if there is no pending result or if the next result is not an update count.

Returns:

update count or -1 if there is no pending result or if the next result is not an update count

Throws:

`QueryTimeoutException` - if the query execution exceeds the query timeout value set and only the statement is rolled back

`PersistenceException` - if the query execution exceeds the query timeout value set and the transaction is rolled back