DBProperties

Adopted By: DBExpression

Declared In: dbkit/properties.h

Protocol Description

An object that conforms to the DBProperties protocol represents a named category of information in an entity (an object that conforms to the DBEntities protocol). Put less formally, a property represents a column in a database table. For example, a table that contains information about a physician's patients might contain the columns ªnameº, ªaddressº, and ªblood typeº. The ªnameº column would be represented as a single property object; similarly, ªaddressº would be a separate property.

While a property object represents a column, it doesn't contain the data that's in the columnÐdata is contained in a table's rows, or *records*. A records is said to have a value *for* a property: A record from the physician's patients table would have a value *for* the blood type property.

A property object describes a column, primarily, through three elements, an entity, a name, and a data type:

- The entity is the object to which the property belongsÐa property can only belong to one entity at a time.
- Within an entity, a property has a unique name such that if two property objects belong to the same entity and have the same name, then they're representing the same category.
- A property's data type establishes the type of data that's held by a record for that propertyÐall values for that property must be of the same type. The type is embodied in a private object that conforms to the DBTypes protocol; the object can be retrieved through the **propertyType** method.

To retrieve a list of properties contained in a particular entity object, you send the entity a **getProperties:** message. You can find a particular property by name by sending the entity a **propertyNamed:** message. The DBProperties that these methods return are created privately by the Database Kit when the entity is read from a model file. You would typically use these properties to initialize a DBBinder, DBRecordList, or DBRecordStream object. Properties are also needed by methods defined by these classes as ªvalue indicesº into records. For example, the DBBinder method **valueForProperty:** returns the DBValue object that's stored in the current record for the given property. Put more naturally, the method returns the value in a particular column.

The DBExpression class adopts the DBProperties protocols. The DBExpression objects that you create and the properties returned by the DBEntities methods described above should suffice for most applicationsÐyou shouldn't need to create your own class that adopts the DBProperties protocol.

Method Types

Identifying a property	- name - setName: - entity
Querying a property	- propertyType - isSingular

- isReadOnly
- isKey
- matchesProperty:

Instance Methods

entity

- (id <DBEntities>)**entity**

Returns the entity to which the property belongs.

isKey

- (BOOL)**isKey**

Returns YES if the property can be used as a *key property* for its entity. A key property is one that can distinguish the records in the entity; in other words, the value of each record for the key property must be unique.

isReadOnly

- (BOOL)**isReadOnly**

Returns YES if the data categorized by the property is read-only; in other words, if it can't be written back to the database.

isSingular

- (BOOL)**isSingular**

Returns YES if the property represents an attribute or a one-to-one relationship. OtherwiseÐto wit, if it's a one-to-many relationshipÐit returns NO.

matchesProperty:

- (BOOL)**matchesProperty:** (id <DBProperties>)*aProperty*

Returns YES if the receiving property and *aProperty* identify the same thingÐif they're in the same entity and have the same name. Otherwise returns NO.

name

- (const char *)**name**

Returns the property's name. For a property read from a model file, this is the name given it by the DBModeler application. To name a DBExpression object, use the **setName:** method.

propertyType

- (id <DBTypes>) **propertyType**

Returns a DBTypes-conforming object that encapsulates the property's data type. All the values that the property categorizes are of this type.

setName:

- (BOOL)**setName:**(const char *)*aName*

Sets the property's name to a copy of *aName*. This method is designed to be used to name DBProperties objects that you create yourself (as explained in the class description, above, such objects will almost certainly be DBExpressions). You shouldn't alter the name of a property that was created for you from a model file. Returns YES if the name was set as requested, otherwise return NO.