DBTypes

Adopted By: no NeXTSTEP classes

Declared In: dbkit/types.h

Protocol Description

The methods in the DBTypes protocol return information about the type of data that's held or described by the object upon which they are invoked. This information doesn't necessarily correspond to an actual valueÐa DBTypes object may not even embody a ªrealº value, and the protocol makes no provision for storing valuesÐit simply provides a means for abstractly describing a data type.

The protocol's two primary methods are **objcType** and **databaseType**; they return strings that represent, respectively, an Objective C data type, and a data type as given in the actual database. The Database Kit uses the following convention in representing Objective C data types as strings:

Objective C type	DBTypes representation
id	ª@º
char *	ª*º
int	ªiº
float	ªfº
double	ªdº

The value returned by **databaseType**, on the other hand, is completely adaptor-dependent. In addition, not all objects have a database type. For example, a relationship that's read from a database model file isn't represented in the actual database, and so will have no database type.

None of the public Database Kit classes implements the DBTypes protocol. However, the kit automatically creates private DBTypes-conforming objects which it uses to store the data types of properties and DBValues. The DBProperties method **propertyType** returns such a private DBTypes object, as does DBValue's **valueType** method.

Method Types

Querying for type	- objcType - databaseType - objcClassName
Comparing types	- isEntity - matchesType:

Instance Methods

databaseType

- (const char *)**databaseType**

Returns a string that represents the object's data type as it resides in the database from which it was read (or to which it will be written).

isEntity

- (BOOL)**isEntity**

Returns YES if the object's data type is an object that conforms to the DBEntities protocol, otherwise returns NO. This method is intended to be used to determine if a property is a relationship, as shown below:

```
if ([[aProperty propertyType] isEntity])
      /* aProperty is a relationship. */
else
      /* aProperty is an attribute. */
```

matchesType:

- (BOOL)**matchesType:**(id <DBTypes>)*anObject*

Returns YES if the object's data type matches that of *anObject*, otherwise returns NO.

objcClassName

- (const char *)**objcClassName**

If the object's type is an **id**, this returns the name of the **id**'s class. If the type isn't an **id**, this returns **nil**.

objcType

- (const char *)**objcType**

Returns a string that represents the object's Objective C data type. The strings that are used by the Database Kit to represent the standard Objective C types are listed in the class description, above.