Gentle.NET ORM

Gentle.NET lets you work with relational databases without using SQL directly.

See this basic example from the Gentle.NET user guide. I have converted it to boo below.

Object-Relational Mapper

Gentle.NET is an object-relational mapper (ORM). ORMs translate your classes and objects into SQL commands for storing those objects and retrieving them from a relational database.

Contrast that with object-oriented databases that store objects directly without using any SQL (like ZODB in Python). See db4objects and Bamboo Prevalence for examples of using those.

How to Use

Use a relational database engine (be it Firebird, MySQL, MSSQL, etc.) to design your tables (see Database Design), and then create the classes that correspond to those tables.

You can also use the free MyGeneration code generation tool to generate those Gentle.NET classes for you automatically (in C#, which you can then convert to boo using SharpDevelop).

Another Alternative: NHibernate

See the NHibernate page for a recipe using the NHibernate ORM. Being based on an existing Java tool, NHibernate probably has or will have a larger user base than Gentle.NET. But so far I like how Gentle.NET uses [attributes] instead of XML to define the mappings between objects and the database (see the example below, although you still need to create a general XML configuration file with info like the connection string).

Gentle.NET Code Sample

I have been experimenting with using Firebird, MyGeneration, and Gentle.NET with boo. I'll post some examples later.

This sample code has not yet been tested, but is provided for illustration:

```
[TableName()]
class User(Persistent):
 _userId as int

 _userName as string

 def constructor(userName as string):
  self(0, userName)

 def constructor(userId as int, userName as string):
  _userId = userId
  _userName = userName

 static def Retrieve(userId as int) as User:
  key = Key(typeof(User), true, 'Id', userId)
  return Broker.RetrieveInstance(typeof(User), key) as User

 [TableColumn('UserId'), PrimaryKey(AutoGenerated: true)]
 Id as int:
  get:
   return _userId
  set:
   _userId = value

 [TableColumn(NotNull: true)]
 Name as string:
  get:
   return _userName
  set:
   _userName = value



//example usage:
ford = User( "Ford Prefect" )
ford.Persist() // save the new user and assign an id value
prefect = User.Retrieve( ford.Id ) // retrieve the existing user
```

Go back to Database Recipes.