GaddioORM

# Summary

GaddioORM is an object-relational mapping project for .NET that allows the user to:

* Avoid writing boilerplate data-access code against a relational database
* Provide simple, object-based access to a relational database, similar to Ruby’s ActiveRecord concept and Microsoft’s Entity Framework.
* Allow developers with little relational database experience to focus on writing business logic code.
* Avoid worrying about hardening code against SQL injection attacks, a common concern when writing dynamic queries.
* Perform complex chains of CRUD functions within transactions.

# Features

## Object-relational mapping

GaddioORM provides a simple mechanism to allow developers to associate a class and its properties to a database table and its columns.

### DBTable attribute

The DBTable attribute defines which table in the database the class represents. This is used by the DA class when generating the opening clause of a dynamic SQL queries to perform basic CRUD functions (Create, Read, Update, Delete).

[DBTable(“Products”)]

public class Product {

}

### DBField attribute

The DBField attribute defines which column a particular class property represents. This is used by the DA class when generating the SET, WHERE, and ORDER BY clauses in dynamic SQL queries.

### DBFieldKeys enum

The DBField.DBFieldKeys enum allows the user to specify column attributes such as whether the column participates in a primary key, whether the column is required to have a value when inserting a record, etc.

[DBField(“ProductId”, DBField.DBFieldKeys.Primary | DBField.DBFieldKeys.AutoIncrement)]

public int Id { get; set; }

## CRUD Functions

The DA class provides a variety of methods that allow the user to perform basic CRUD operations without writing boilerplate queries, whether embedded in C# code or as stored procedures.

### Get<T>

The Get and GetList family of functions allows the user to retrieve one or more objects of a specified type T.

Example:

Product proto = new Product() {

Id = 71

}

Product result = DA.Get<Product>(proto, “Id”);

The DA class generates the following SQL command under the hood and executes it against the database.

SELECT \*

FROM Products

WHERE ProductId=@ProductId

Note: The @ProductId variable is assigned the value of 71 using Microsoft’s SqlCommand parameter interface, which provides protection against SQL injection attacks.