Writing database queries for an Application is always boring and error prone. Came across [Vega](https://github.com/aadreja/vega/) – one of the fastest .net ORM (Object-relational Mapping).

Following are the exciting features of Vega: -

* Inbuilt Row Versioning
* Powerful Audit Trail to keep track of all changes
* Object Mapping via Emit which are cached to get performance same as manual object mapping.
* Inbuilt Paging without offset is fastest of all. Supports paging with offset as well.
* Concurrency check during Update and Delete.
* Inbuilt implementation of common fields like CreatedBy, CreatedOn, UpdatedBy, UpdatedOn, IsActive, VersionNo
* Define Virtual Foreign Keys to check data integrity on Delete.
* Cross database support for Microsoft SQL Server, PostgreSQL, SQLite

# **Why Vega?**

Vega is one of the fastest cross platform .net ORM. Below is the performance matrix comparing ADO vs Vega

## **Performance Results for 1000 records 5 iteration**

| **Run** | **1** | **2** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- | --- |
| **Insert Tests** |  |  |  |  |  |
| - ADO | 135ms | 126ms | 121ms | 142ms | 151ms |
| - Vega | 212ms | 171ms | 148ms | 177ms | 193ms |
| **Update Tests** |  |  |  |  |  |
| - ADO | 140ms | 143ms | 483ms | 157ms | 163ms |
| - Vega | 159ms | 163ms | 351ms | 173ms | 162ms |
| **Select Tests** |  |  |  |  |  |
| - ADO | 104ms | 106ms | 287ms | 136ms | 133ms |
| - Vega | 133ms | 109ms | 159ms | 143ms | 137ms |
| **Select List Tests** |  |  |  |  |  |
| - ADO | 5ms | 3ms | 3ms | 3ms | 3ms |
| - Vega | 7ms | 4ms | 5ms | 3ms | 4ms |

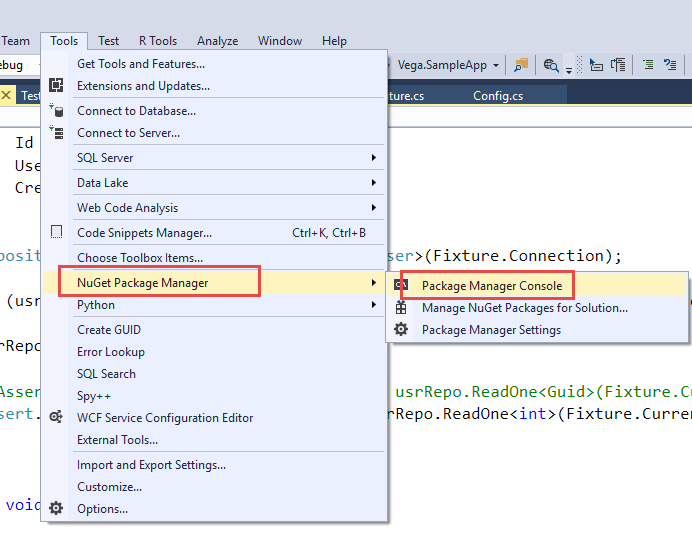
# **How to get Vega?**

There are following ways to install & use vega.

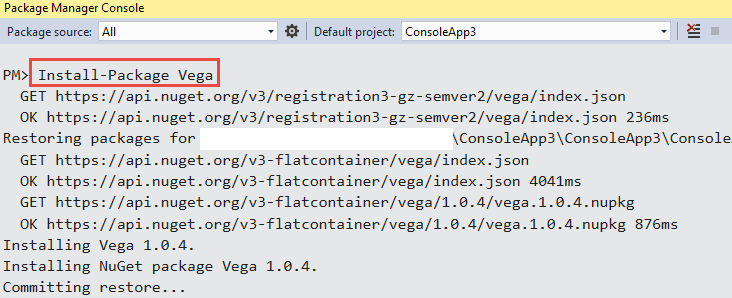
## Install from Nuget

To get the .NET 4.0 or .NET Standard version of the Vega from Nuget

* 1. Go to Visual Studio -> Tools -> NuGet Package Manager -> Package Manager Console



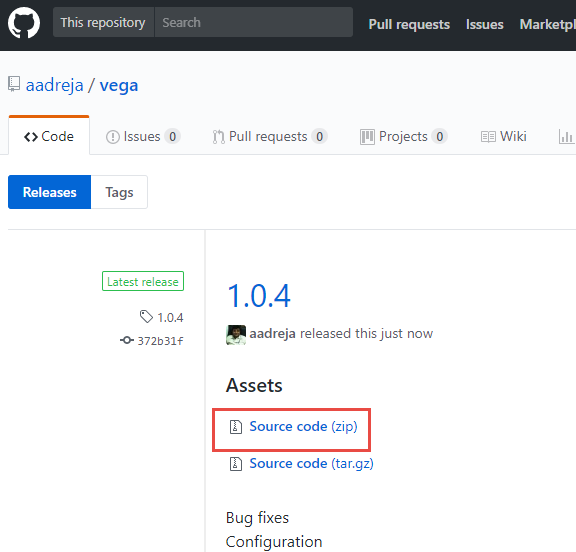
* 1. In Package Manager Console type “Install-Package Vega” and hit Enter.



## Download source code from GitHub

To get the source code from GitHub

* 1. Go to <https://github.com/aadreja/vega/releases>
  2. Download the latest release



* 1. Unzip source code, add to solution & reference to your project.

# **How to use Vega?**

Vega is simple to use. Following are steps to begin: -

1. Onetime startup/optional configuration

By default vega works with default configuration. You may override default configuration in Configuration class. It has to be done once and all settings will be cached till application life.

//Configure Vega

Configuration configuration = new Configuration

{

CreatedUpdatedByColumnType = DbType.Int32

};

Config.Configure(configuration);

1. Create an Entity class derived from Vega.EntityBase

Example: -

[Table(NeedsHistory = true)]

public class Country : EntityBase

{

[PrimaryKey(true)]

public int Id { get; set; }

public string Name { get; set; }

public string ShortCode { get; set; }

}

1. Perform CRUD operations
   1. Insert

Simply create Entity & set properties for Insert. Create a connection object, pass it to Repository & call Add method which will return Id of newly create record

Country country = new Country()

{

Name = "India",

ShortCode = "IN",

CreatedBy = 1 //user id created record

};

using (SqlConnection connection = new SqlConnection(conString))

{

Repository<Country> countryRepo = new Repository<Country>(connection);

int result = (int)countryRepo.Add(country);

}

* 1. Update

To update record use Update method of repository.

City city = new City()

{

Id = 1,

Name = "Ahmedabad",

State = "GU",

CountryId = 1,

Longitude = 102.23m,

Latitude = 124.23m,

UpdatedBy = 1

};

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

bool result = cityRepo.Update(city);

}

* 1. Delete

To delete record use Delete method of repository. Soft Delete if Entity contains IsActive column.

//soft delete

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

bool result = cityRepo.Delete(1, 1); //Record Id & User Id

}

//hard delete

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

bool result = cityRepo.HardDelete(1, 1); //Record Id & User Id

}

* 1. Read One

Reads first record with specified criteria & converts it to object.

//read one with specified id

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

City city = cityRepo.ReadOne(1); //Record Id

}

//read one with specified dynamic criteria

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

City city = cityRepo.ReadOne("Name=@Name", new { Name = "Ahmedabad" }); //Dynamic criteria

}

* 1. Read All

Reads all records with specified criteria & converts it to enumerable object which can be converted to List.

//read all records

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

List<City> cityList = cityRepo.ReadAll().ToList();

}

//read all with specified dynamic criteria

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

List<City> cityList = cityRepo.ReadAll("Name=@Name", new { Name = "Ahmedabad" }).ToList(); //Dynamic criteria

}

* 1. Read Paged

Reads all records with server side paging, specified criteria & converts it to enumerable object which can be converted to List.

//read all records (name column) of page no 1 with page size 5

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

List<City> cityList = cityRepo.ReadAllPaged("name", 1, 5).ToList();

}

//read all records (name column) of page 1 with page size 5 & specified dynamic criteria

using (SqlConnection con = new SqlConnection(connectionString))

{

Repository<City> cityRepo = new Repository<City>(con);

List<City> cityList = cityRepo.ReadAllPaged("name", 1, 5, null, "State=@State", new { State = "GU" }).ToList(); //Dynamic criteria

}