This repository ▾ | Search or type a command ⑦   Explore Gist Blog Help   rofr ✚▾ ⚒ ⏏

DevrexLabs / **OrigoDB.Workshop**

👁 Unwatch ▾ | 1   ★ Star | 0   ⑂ Fork | 0

04 Hosting

[Edit] [New Page]

rofr edited this page 3 hours ago · 5 commits

This module covers the different options hosting the engine and in-memory model.

Exercise 1

In this exercise you will create a skeleton ASP.NET MVC application hosting the Todo.Core model. We will have VS generate some CRUD scaffolding that we can use as a starting point. For this purpose we need some fake classes which will be deleted once the scaffolding has been generated.

1. Open the Todo solution located in the 'Ex01' folder.
2. Add an MVC application to the solution called `Todo.Mvc`
3. Add the following classes somewhere in the project, note the namespace!

```
namespace Todo.Core
{
    public class Todo
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public DateTime? Due { get; set; }
        public DateTime? Completed { get; set; }
    }

    public class TodoContext : DbContext
    {
        public DbSet<Todo> Todos { get; set; }
    }
}
```

4. Build the solution
5. Add an MVC controller with read/write actions and views, using Entity Framework. Choose the Todo and TodoContext classes. (Right click controllers in solution explorer)
6. Add a reference to the Todo.Core project
7. Modify the `Core.TodoView` class to include the Due and Completed fields
8. In the generated controller, replace the TodoContext field with the following: `csharp`
   `IEngine<TodoModel> engine = Engine.For<TodoModel>();`
9. Change the model type of all the generated views from `Todo` to `TodoView`
10. Make all the controller methods work, modifying the core library and views as needed.

[+ Add a custom footer]

▾ Pages (8)

01 Introduction

02 Modeling

03 Testing

04 Hosting

05 Configuration

06 Immutability

07 Server

Home

[+ Add a custom sidebar]

Clone this wiki locally

https://github.com/DevrexL 📋

🖥 **Clone in Desktop**