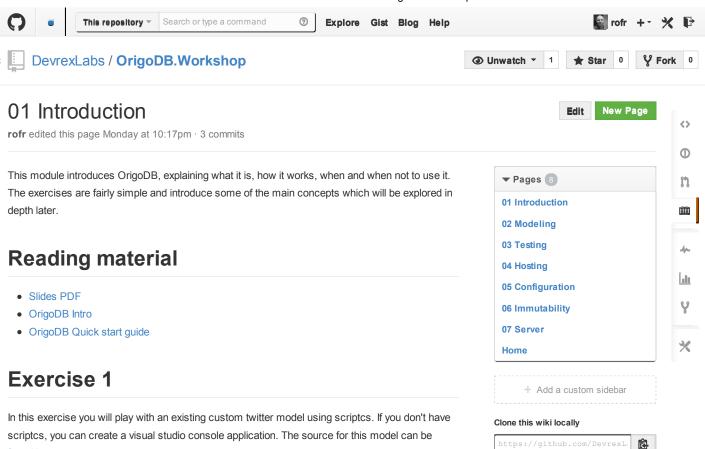
found here



Clone in Desktop

- 1. Start a command prompt and navigate to the modules [Ex01] folder
- 2. Install the OrigoDB from nuget by typing <code>scriptcs -install origodb.core</code>
- 3. Start the scriptcs REPL by typing scriptcs and enter the following statements:

```
#r Twitter.Core.dll
using OrigoDB.Core;
using Twitter.Core;
var t = new TwitterVerse();
t.AddUser("unclebob");
t.AddUser("robtheslob");
t.Follow("robtheslob", "unclebob")
t.Tweet("unclebob", "If you don't do TDD ur doing it wrong", DateTime.Now)
t.Unfollow("unclebob", "robtheslob")
```

You have created and manipulated a data model instance in memory but there is no engine controlling concurrency or providing persistence. You will fix this in the next step. Press CTRL+C to exit scriptcs, the current twitterverse will be lost forever.

- 4. Install origodb.core from nuget
- 5. Create a text file called twitter.csx with the following content:

```
#r Twitter.Core.dll

using OrigoDB.Core;
using Twitter.Core;

var db = Db.For<TwitterVerse>(); //transparent proxy with backing file storage in curred.

db.AddUser("robtheslob");
db.Follow("neo");
db.Follow("neo", "robtheslob");
db.Tweet("neo", "What is the Matrix?", DateTime.Now);
```

- 6. Start the script and enter interactive REPL mode using $\[$ scriptcs twitter.csx -REPL $\[$
- 7. Exit with CTRL+C when you've finished playing around

Takeways

- · A snapshot and journal was created on disk
- If you execute Db.For<TwitterVerse>() again, the in-memory model will be restored from disk
- You used the transparent proxy feature
- · Scriptcs is awesome

Exercise 2

In this excercise you will create a custom data model using object oriented domain modeling.

- 1. Create an empty NET 4.0 Class Library Project using Visual Studio named Todo.Core in Ex02\Starter
- 2. Create an entity class called Todo representing a single todo item.

```
[Serializable]
public class Todo
{
    public DateTime? Due { get; private set; }
    public DateTime? Completed { get; private set; }
    public string Title { get; private set; }
    public readonly int Id;

    public Todo(int id, string title = "Untitled")
    {
        Id = id;
        Title = title;
    }
}
```

- 3. Add a nuget reference to OrigoDB.Core
- 4. Create an anemic root data model class deriving from OrigoDB.Core.Model

```
[Serializable]
public class TodoModel : Model
{
    public SortedDictionary<int, Todo> Todos { get; private set; }
    int _nextId = 1;

    public int GetNextId()
    {
        return _nextId++;
    }

    public TodoModel()
    {
        Todos = new SortedDictionary<int, Todo>();
    }
}
```

5. Create the following commands:

```
[Serializable]
public class AddTodoCommand : Command<TodoModel, int>
{
    public readonly string Title;

    public AddTodoCommand(string title)
    {
        Title = title;
    }

    public override int Execute(TodoModel model)
    {
        int id = model.GetNextId();
        model.Todos[id] = new Todo(id, Title);
        return id;
    }
}
```

```
[Serializable]
public class SetCompletedCommand : Command<TodoModel>
{
    public readonly int Id;
    public readonly DateTime Completed;

    public SetCompletedCommand(int id, DateTime completed)
    {
        Id = id;
        Completed = completed;
    }

    public override void Execute(TodoModel model)
    {
        if (!model.Todos.ContainsKey(Id)) Abort("No such todo");
        model.Todos[Id].SetCompleted(Completed);
    }
}
```

6. Add a SetCompleted method to the Todo class and make sure everything builds

Now it's time to test drive the model, let's use scriptcs again.

7. Create a text file named todo.csx in the Ex02 directory with the following contents:

```
#r Todo.Core/bin/debug/Todo.Core.dll

using Todo.Core;
using OrigoDB.Core;

var engine = Engine.LoadOrCreate<TodoModel>();
var command = new AddTodoCommand("Learn Scala");
int id = engine.Execute(command);

var todo = engine.Execute(db => db.Todos.Values.Where(t => t.Id == id).Single());
engine.Execute(new SetCompletedCommand(id, DateTime.Now));

var todo2 = engine.Execute(db => db.Todos[id]);
```

- 8. Start the script and remain in REPL mode.
- 9. Examine existing variables, play around, ask questions and discuss

Review

- You have created a class library with an entity, a root data model and commands.
- You have created an engine and executed commands and ad-hoc queries
- Note the snapshot and journal
- Scriptcs is awesome

```
→ Add a custom footer
```

© 2014 GitHub, Inc. Terms Privacy Security Contact

Status API Training Shop Blog About