OrigoDB Workshop

Module 2 - Modeling

Module overview

- Data models
- Commands
- Queries
- Labs

Creating a data model

- Domain specific vs. Generic vs. Hybrid
- Rich vs. Anemic
- Explicit vs. Implicit commands
- References vs. Foreign keys
- Open vs. Closed

Generic models

- Generic models
  - Graph
  - Relational
  - Key/Value, Document
  - Origodb.Models.Redis

Other model types

- Interpreter hosting – Javascript, Roslyn
- Lucene index
- Machine learning models (Accord.NET)

Object oriented domain modeling

- DDD?
- Entities and collections
- Choose collections wisely – space/time tradeoffs
- Transaction script pattern or thin commands
- Avoid CRUD

Silly Relational

```
[Serializable]
public class RelationalModel : Model
{
    private DataSet _dataset;

    public RelationalModel()
    {
        _dataset = new DataSet();
    }
   //... ExecuteQuery and ExecuteCommand omitted
}
```

Generic Relational

```
[Serializable]
public class RelationalModel : Model
{
    Dictionary<string,SortedDictionary<object>> _tables;
}
```

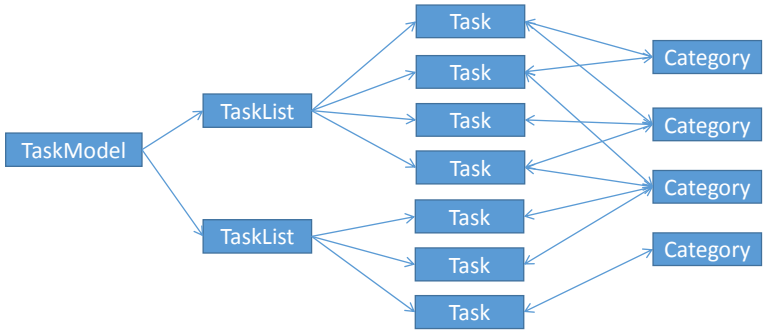
Domain specific relational

```
[Serializable]
public class Model : Model
{
    SortedDictionary<int,Customer> _customers;
    SortedDictionary<int,Order> _orders;
    SortedDictionary<int,Product> _products;
    SortedDictionary<string,Customer> _customersByName;
}

[Serializable]
public class Order {
   public int CustomerId; //foreign key vs. reference
}
```

Interpreter hosting

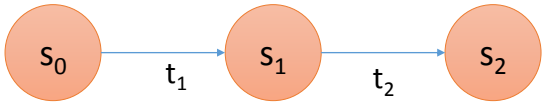
```
[Serializable]
public class JurassicModel : Model
{
    private ScriptEngine _scriptEngine;

    public JurassicModel()
    {
        _scriptEngine = new ScriptEngine();
        _scriptEngine.Execute("var model = {}");
    }
   //... ExecuteQuery and ExecuteCommand omitted
}
```

The model is an object graph



Commands

- Serial execution
- Exclusive access to the model
- Transition the model from one valid state to the next



Command guidelines

- No side effects or external actions
- No external dependencies
- Unhandled exceptions trigger rollback (full restore)
- Call Command.Abort() to signal exception

Query alternatives

- Ad-hoc lambda: Engine.Execute(Func<M,R> query)
- Derive from Query<M,R>
- Compiled LINQ: Engine.Execute<R>(string)

Query guidelines

- Know thy object graphs
- Don’t modify the model

Lab

- Extending the Todo model with Categories, a many to many relationship
- Querying with lambdas
- Adding Commands, Queries and Views