.NET
Applying Domain-Driven Design
Jimmy Nilsson
About Jimmy Nilsson

- Primarily a developer, but also a trainer and author
- Blogs at www.jnsk.se/weblog/
- Author of "Applying Domain-Driven Design and Patterns" and ".NET Enterprise Design"
Main Focus of DDD?

- The core is the main focus!
- Forget distractions!
Relational database
Why DDD?

- Knowledge-rich design
- Mapping to domain
- Reduced complexity
- Testability
- Maintainability
- Etc, etc...

©ØREDEV
Problem Description

• A complex, large scale and long lived order application will be built

• Some very simple examples of requirements:
  • Order has a customer, customer has orders
  • Order has one or more lines
  • Order has a value
  • Customer has address
  • ...

© ReDeV
Transaction Script [PoEAA]

TransactionScriptA
  +RunScript1()
  +RunScript2()

TransactionScript
  +Run()

ConcreteTransactionScriptA
  +Run()

ConcreteTransactionScriptB
  +Run()
Table Module [PoEAA]

- **TableModule**
  - +Process1(in id)
  - +Process2()

- **RecordSet**

Relationship:
- 1
Domain Model [PoEAA]
• Because of the problem description, we choose Domain Model

• Now we need a Domain Model style...
Value Objects [DDD]

Customer
+CustomerId
+Name
+Street
+City
+State

Customer
+CustomerId
+Name

Address
+Street
+City
+State
Aggregates [DDD]

- Engine
- Car
- Customer
- Wheel
- Position
- Tire

Relationships:
- 1:1 Car to Engine
- 1:4 Wheel to Car
- 1:4 Tire to Car
- * Position to Wheel
Repositories [DDD]

client ➔ repository ➔ database

selection criteria ➔ matching objects

Delegate ➔
Factories [DDD]

- `client` → `FACTORY` with `new(parameters)` → `product`
- `FACTORY` → `product` with `Create`
What problems remain?

- Infrastructure...

- Assumption 1: Relational database
  - How?
  - Assumption 2: O/R Mapping
  - Which?

- Assumption 3: *For eg* NHibernate
NHibernate

• Domain Model style: POCO / Persistent Ignorant (PI)
• Mapper style: Framework
• Start with: Domain Model, Tables, Metadata
• API-Focus: Domain Model
• Query language: String-based, Query Object-based
• Open source, production version
• Many databases are supported
• Port of Hibernate 2.0.3
Data Mapper [PoEAA]

• For example Customer.hbm.xml

• `<class name="Customer" table="Customers">
  • `<id name="Id" access="field">
    • `<unsaved-value="00000000-0000-0000-0000-000000000000">
      • `<generator class="guid.comb"/>
  • </id>
• `<property name="Name"
  access="field.camelcase-underscore"/>

ØREDEV
CRUD – C

• _session.Reconnect();

• try
  • {
  •   _session.SaveOr Update (customer);
  •   _session.Flush();
  •  }
• finally
  • {
  •   _session.Disconnect();
  •  "}


CRUD – R

• (Customer)_session
• .Load(typeof(Customer), id);
References

• [DDD] Eric Evans; *Domain-Driven Design*
• [PoEAA] Martin Fowler; *Patterns of Enterprise Application Architecture*
• [NHibernate] nhibernate.sourceforge.net
• [ADDDP] Jimmy Nilsson; *Applying Domain-Driven Design and Patterns*
Remember!

Enter the evaluation form and be a part of making Øredev even better.

You will automatically be part of the lottery tomorrow