

Module 10: "Unit of Work"



TEKNOLOGISK
INSTITUT

Agenda

- ▶ Introductory Example: Adding Comments to Products
- ▶ Challenges
- ▶ Pattern: Unit of Work
- ▶ Overview of Unit of Work
- ▶ Discussion

Introductory Example: Products and Comments

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Manufacturer { get; set; }
    public Category? Category { get; set; }
    public virtual ICollection<Comment> Comments { get; set; }
}
```

```
public class Comment
{
    public int Id { get; set; }
    public Product Product { get; set; }
    public string Description { get; set; }
}
```

Challenges

- ▶ Do we really need to retrieve the comments for each product individually?
- ▶ How do we even update the repositories?
 - Do we need a **Save()** or **Commit()** on each of the repositories?
- ▶ How do we handle updates as a single "unit"?
 - Consistent
 - Handling concurrency problems
 - "Transactional"

First Things First...

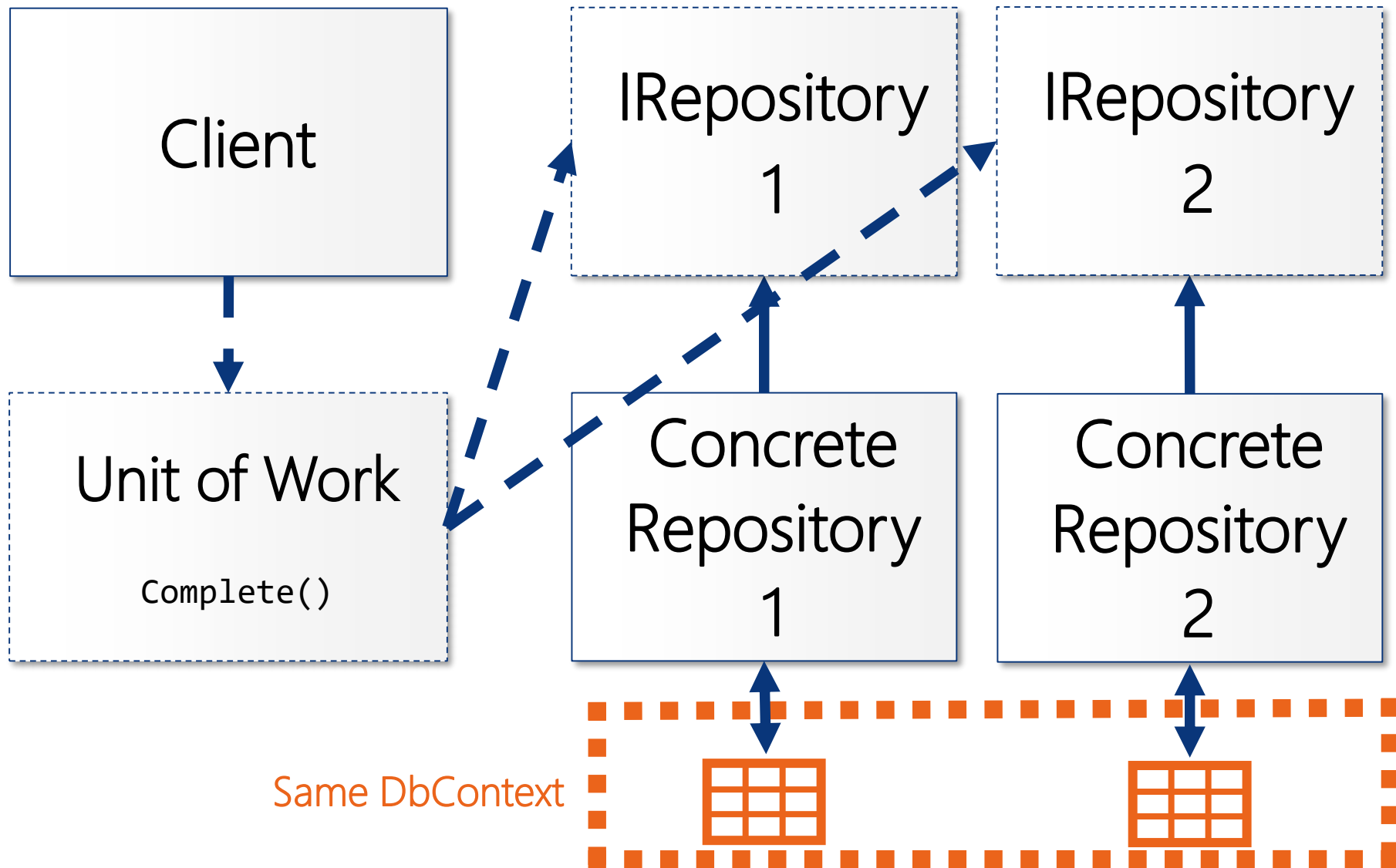
- ▶ Eager-loading of related entities can be handled using Repository itself – without Unit of Work 😊

```
class ProductRepository : Repository<Product>, IProductRepository
{
    ...
    public IEnumerable<Product> GetAllWithComments() =>
        ProductsContext.Products
            .Include(product => product.Comments)
            .Where(product => product.Comments.Any())
            .ToList()
            ;
}
```

Pattern: Unit of Work

- ▶ *Maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems*
- ▶ Outline
 - Encapsulate data access to multiple repositories as a single indivisible unit
 - Essentially, Entity Framework contexts are almost already such units
- ▶ Origin:
 - Martin Fowler (2003)

Overview of Unit of Work Pattern



Overview of Unit of Work Pattern

- ▶ Client
 - Queries and updates data through the Unit of Work Interface
 - Only know the Unit of Work Interface and general Repository interfaces
- ▶ Unit of Work
 - Interface or base class exposing in a persistence-aware update
 - Coordinates transactional saves, concurrency, and consistency
 - Very simple implementation
- ▶ Repository 1 and 2
 - General Repository interfaces
 - Concrete implementations provides persistence-dependent data access code for a specific concrete data source

Discussion

- ▶ Unit of Work with just a single Repository
 - Sometimes merged
 - Source of confusion in literature



WINCUBATE

Jesper Gulmann Henriksen

PhD, MCT, MCSD, MCPD

Phone : +45 22 12 36 31

Email : jgh@wincubate.net

WWW : <http://www.wincubate.net>

Ringgårdsvej 4A

8270 Højbjerg

Denmark