Repositories

Another Important Pattern for Managing Domain Complexity

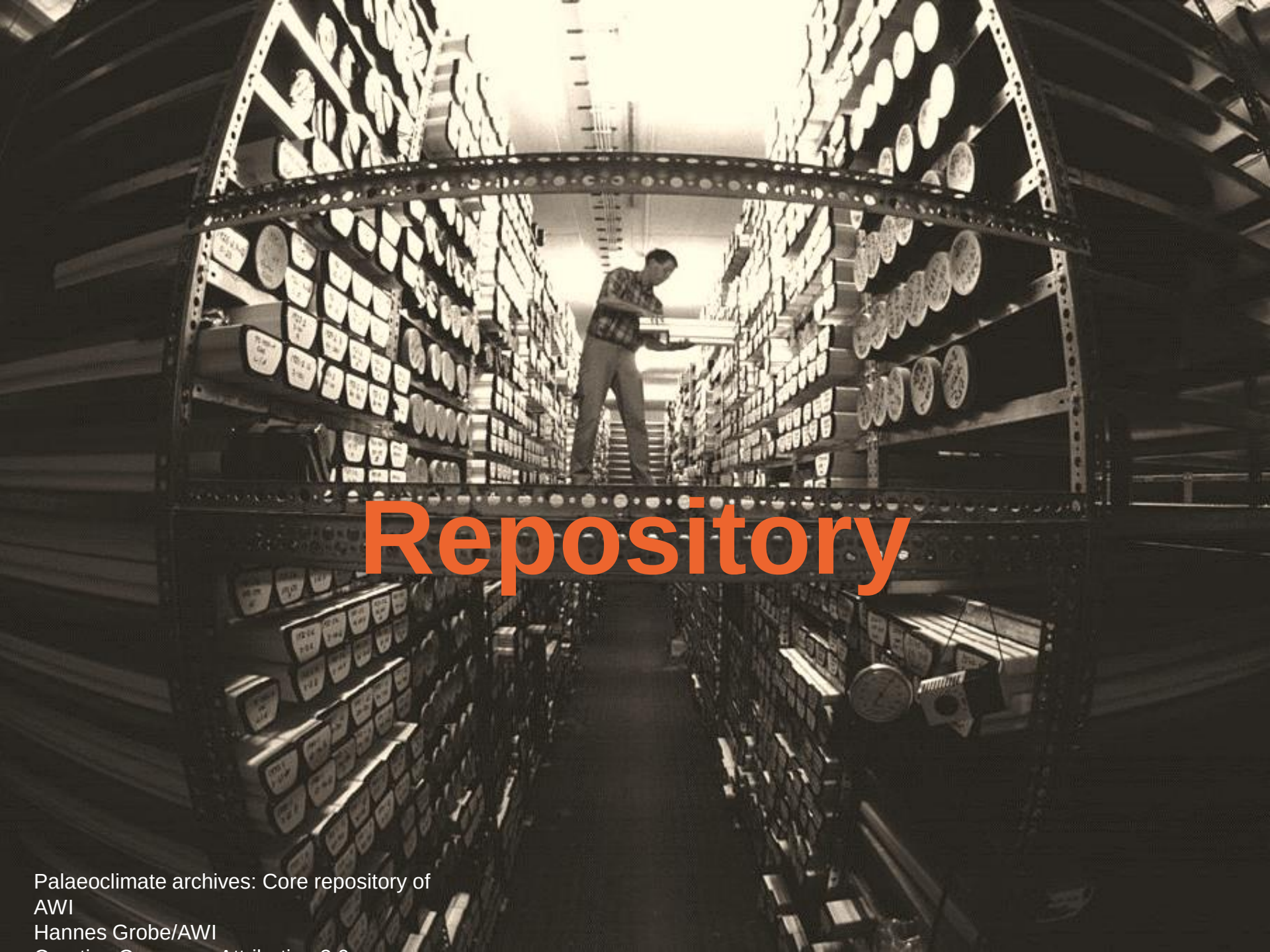
Julie Lerman
TheDataFarm.com
@julielerman

Steve Smith
Ardalis.com
@ardalis

pluralsight
hardcore dev and IT training

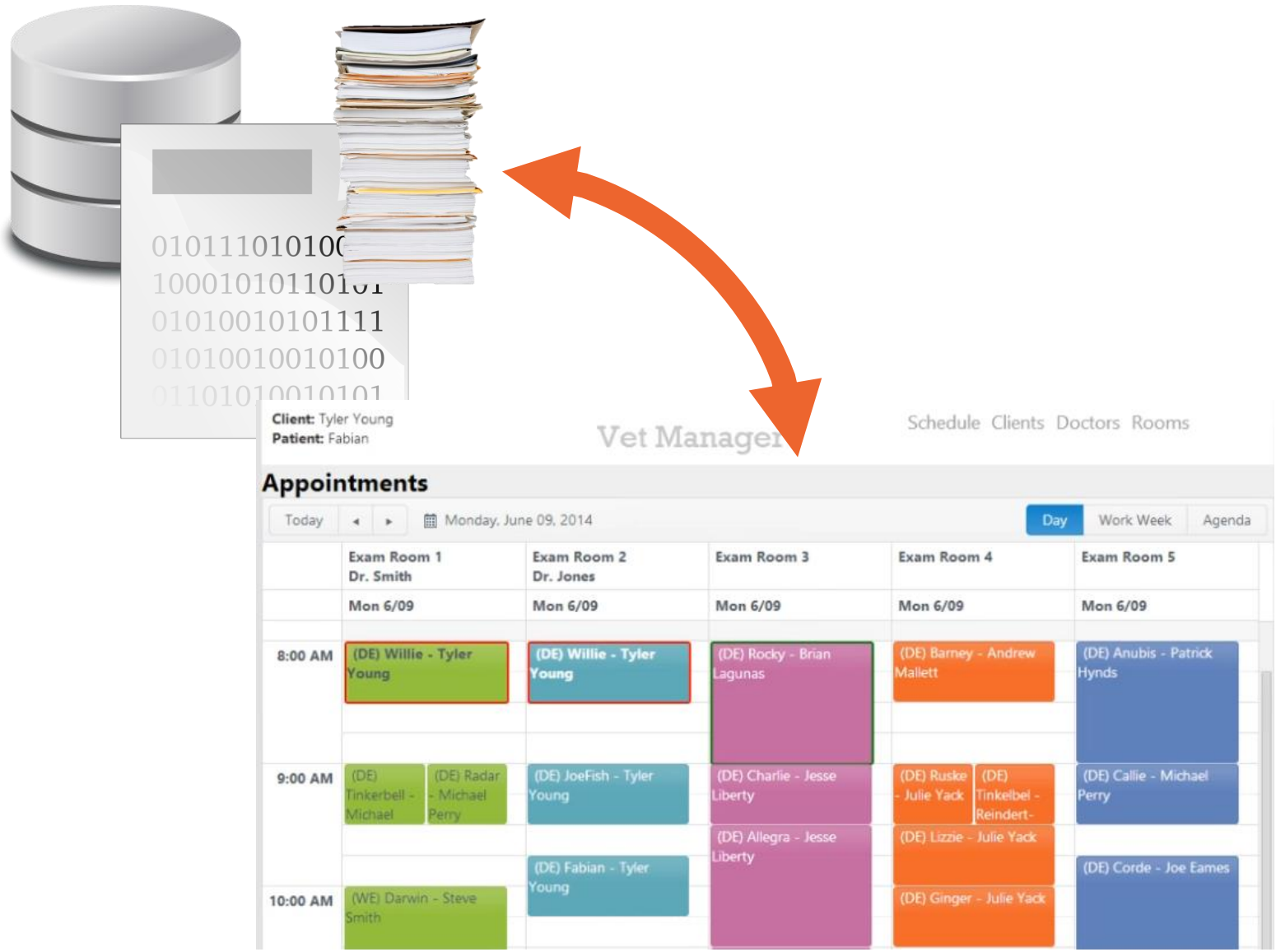
In This Module

- Define Repositories
- Tips & Benefits
- Common Repository Conundrums
- Repository Implementations in our App

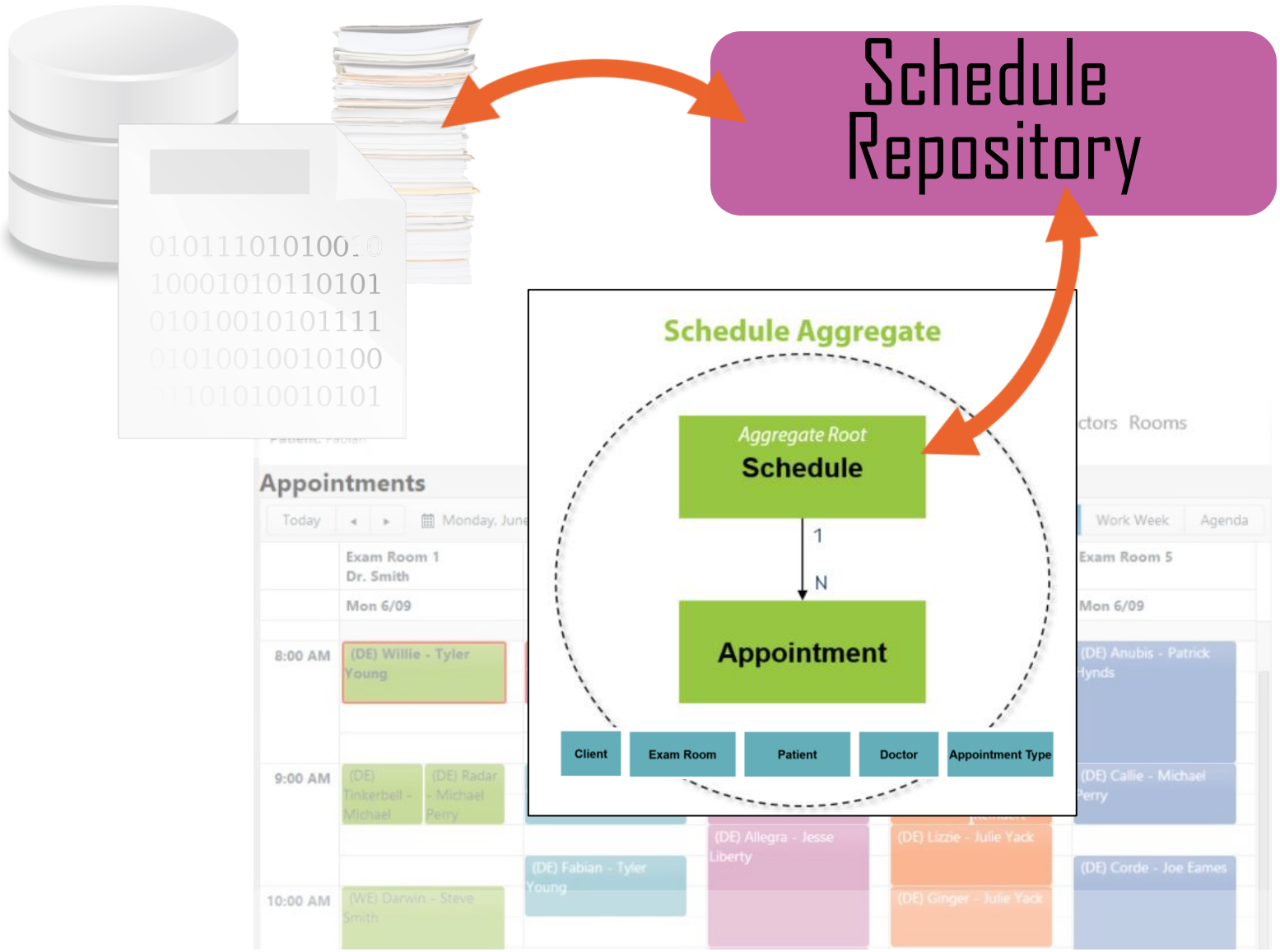


Palaeoclimate archives: Core repository of AWI
Hannes Grobe/AWI

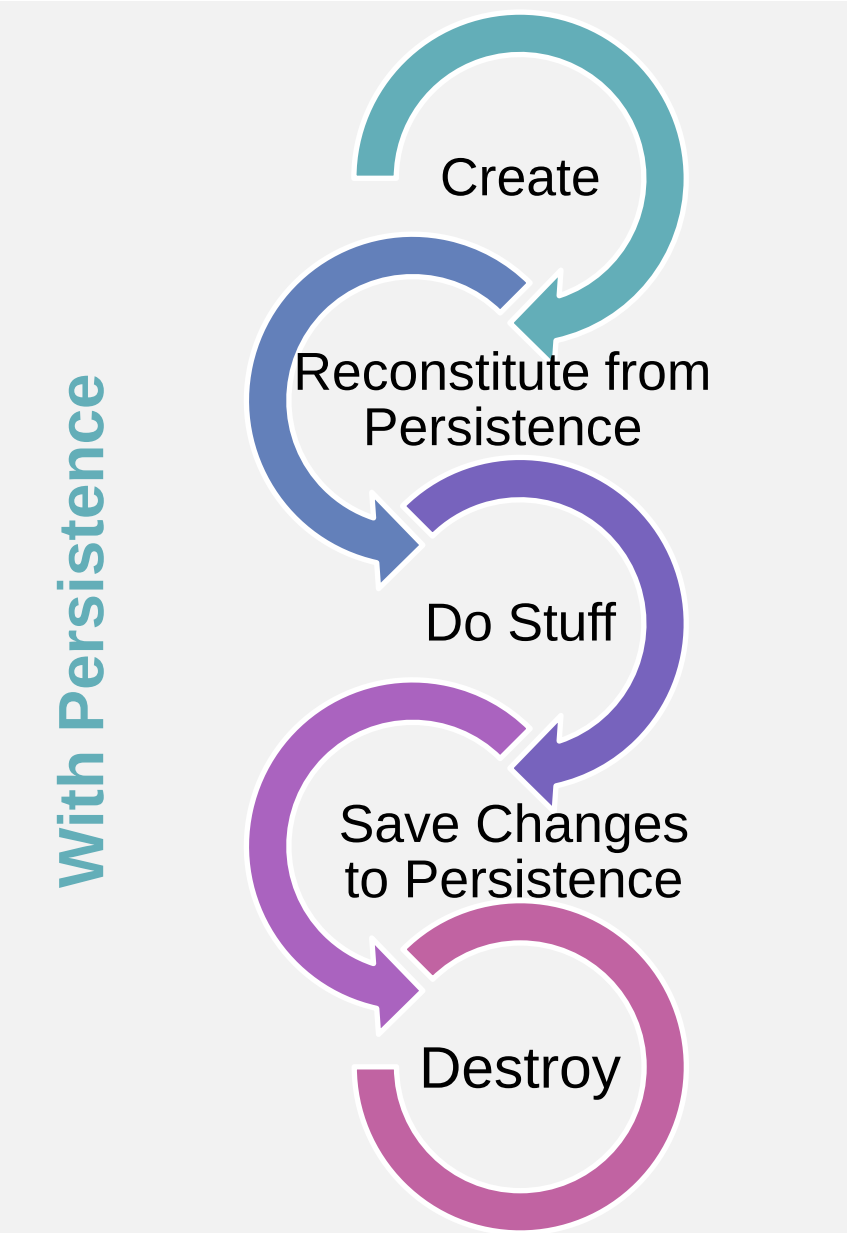
Retrieving Objects



Retrieving Objects



Object Life Cycles



> A repository represents all objects of a certain type as a conceptual set…like a collection with more elaborate querying capability.

— **Eric Evans**
Domain-Driven Design

Repository Tips

Implement a

```
public Schedule GetScheduledAppointmentsForDate(int clinicId, DateTime date) {
  var scheduleGraph = QueryScheduleForThisOffice(clinicId)
                      .Select(s => new
                      {
                          Schedule = s,
                          Appointments = s.Appointments
                              .Where(a =>
                              DbFunctions.DiffDays(date, a.TimeRange.Start) == 0
                              )
                      })
                      .SingleOrDefault();
  var schedule = scheduleGraph.Schedule;
  schedule.DateRange = new DateTimeRange(date, date.AddDays(1));
  return schedule;
}
```

Object selection

roots

on persistence

Repository Benefits

- Provides common abstraction for persistenc
- Promotes Separation of Concerns
- Communicates Design Decisions
- Enables Testability
- Improved Maintainability

Client code can be ignorant of repository implementation

…but developers cannot

Common Repository Blunders		
N+1 Query Errors	Inappropriate use of eager or lazy loading	Fetching more data than required

Repositories

Factories

objects

Factories create new objects
Repositories find and update existing objects
A Repository can use a Factory to create its objects

persistence

Factories : no, no, no
Repositories: yes, yes, yes

To IRepository<T> or Not To IRepository<T> ?

```
public interface IRepository<TEntity> where TEntity : IEntity
{
    IEnumerable<TEntity> List();
    TEntity GetById(int id);
    void Insert(TEntity entity);
    void Update(TEntity entity);
    void Delete(int id);
}
```

```
public interface IScheduleRepository
{
    Schedule GetScheduledAppointmentsForDate(int clinicId, DateTime date);
    void Update(Schedule schedule);
}
```

Generic Repositories in DDD

```
public class Repository<TEntity> : IRepository<TEntity> where TEntity : class, IEntity
{
    private readonly CrudContext _context;
```

```
public class NonRoot : IEntity
{
    public int Id...
}

public class ClientCode
{
    public void Foo()
    {
        var result = new Repository<NonRoot>().GetById(1);
    }
}
```

```
    }

    public void Delete(int id)
    {
        var entityToDelete = _dbSet.Find(id);
        _dbSet.Remove(entityToDelete);
        _context.SaveChanges();
    }
}
```

Generic Repositories in DDD

```
public interface IAggregateRoot : IEntity { }

public class Root : IAggregateRoot
{
    public int Id...
}
```

```
public class Repository<TEntity> : IRepository<TEntity> where TEntity : class, IAggregateRoot
{
```

```
public class ClientCode
{
    public void Foo()
    {
        var result = new Repository<NonRoot>().GetById(1);
    }
}
```

class ClientPatientManagement.Data.NonRoot

C#: This argument type is not within its bounds

Repositories in our Application

Glossary of Terms from this Module

Repository

A class that encapsulates the data persistence for an **aggregate root**

ACID

Atomic, Consistent, Isolated, and Durable

References

Books

Domain-Driven Design http://amzn.to/1kstiRg
Implementing Domain-Driven Design http://amzn.to/1dgYRY3

Web

Eric Evan's website DomainLanguage.com
DDD Community website DDDCommunity.org

On Pluralsight:

Entity Framework in the Enterprise – bit.ly/PS-EFEnterprise
SOLID Principles of OO Design - bit.ly/solid-smith

Thanks!

Julie Lerman

TheDataFarm.com

Twitter: @julielerman

Steve Smith

Ardalis.com

Twitter: @ardalis

To Teach Is To Learn Twice

pluralsight
hardcore developer training