Domain Modeling Best Practices

Vladimir Khorikov

@vkhorikov www.enterprisecraftsmanship.com

The YAGNI Principle

You aren't going to need it

Business requirements change constantly

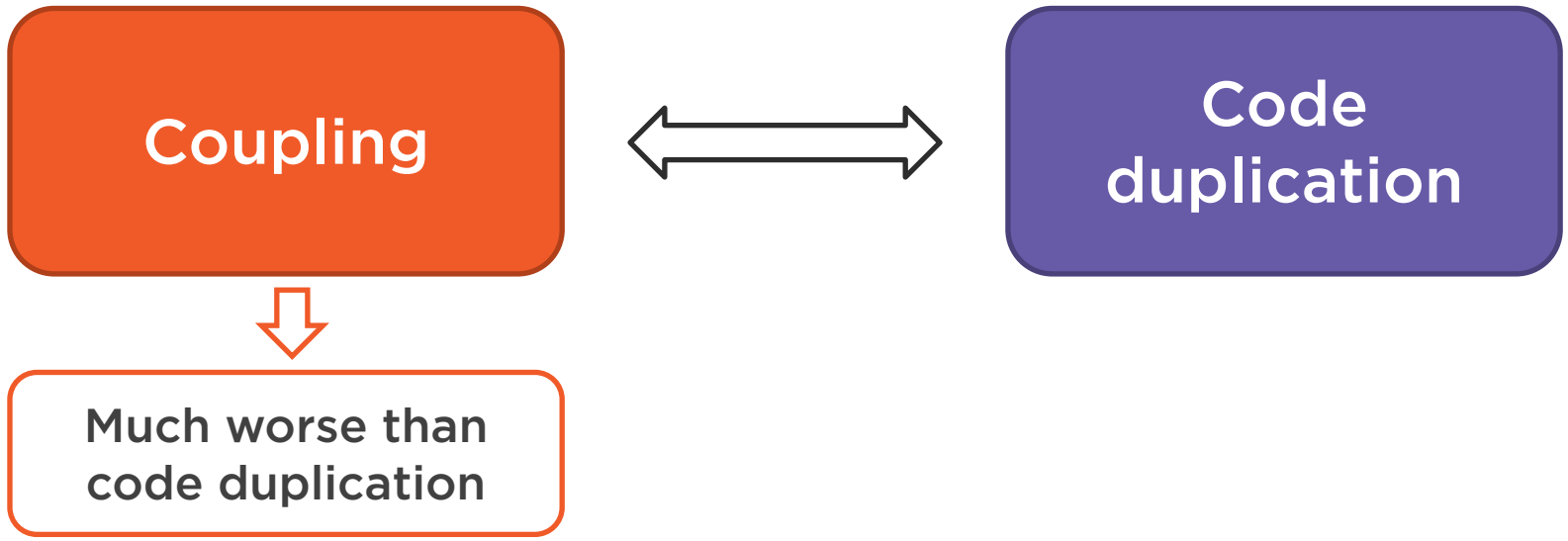
Your code is not an asset, it's a liability

Build as specific solution to your problem as possible

Don't use a boilerplate set of domain classes

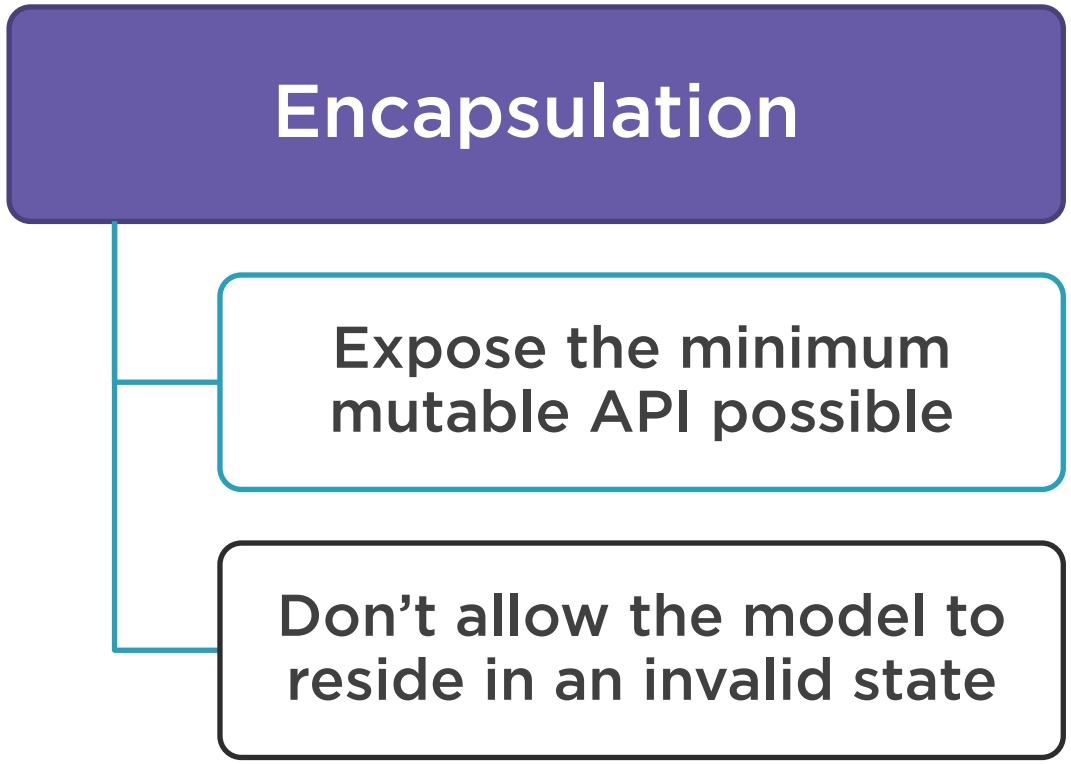
Sharing Domain Logic Between Projects

- The DRY principle is for a single bounded context only
- Bounded contexts have different perspectives on same concepts



- Sharing utility code is mostly OK
- Bounded contexts should always evolve independently

Domain Model Encapsulation



Domain Model Isolation

Separating domain logic from external influence

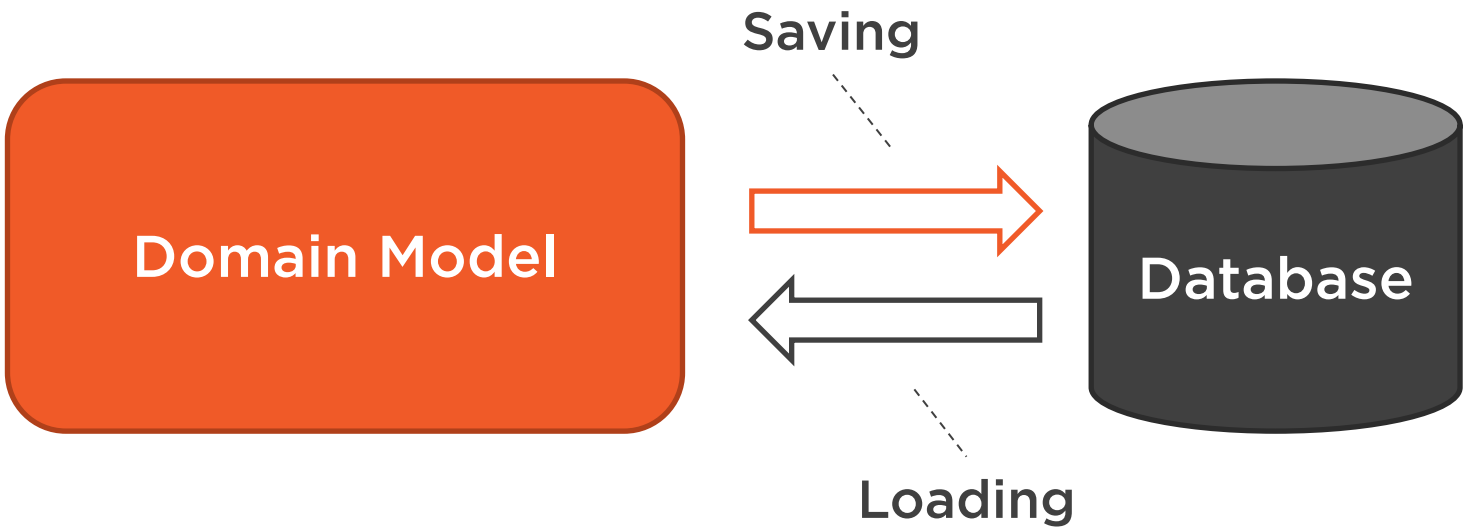
Isolation

Separation of concerns

Encapsulation

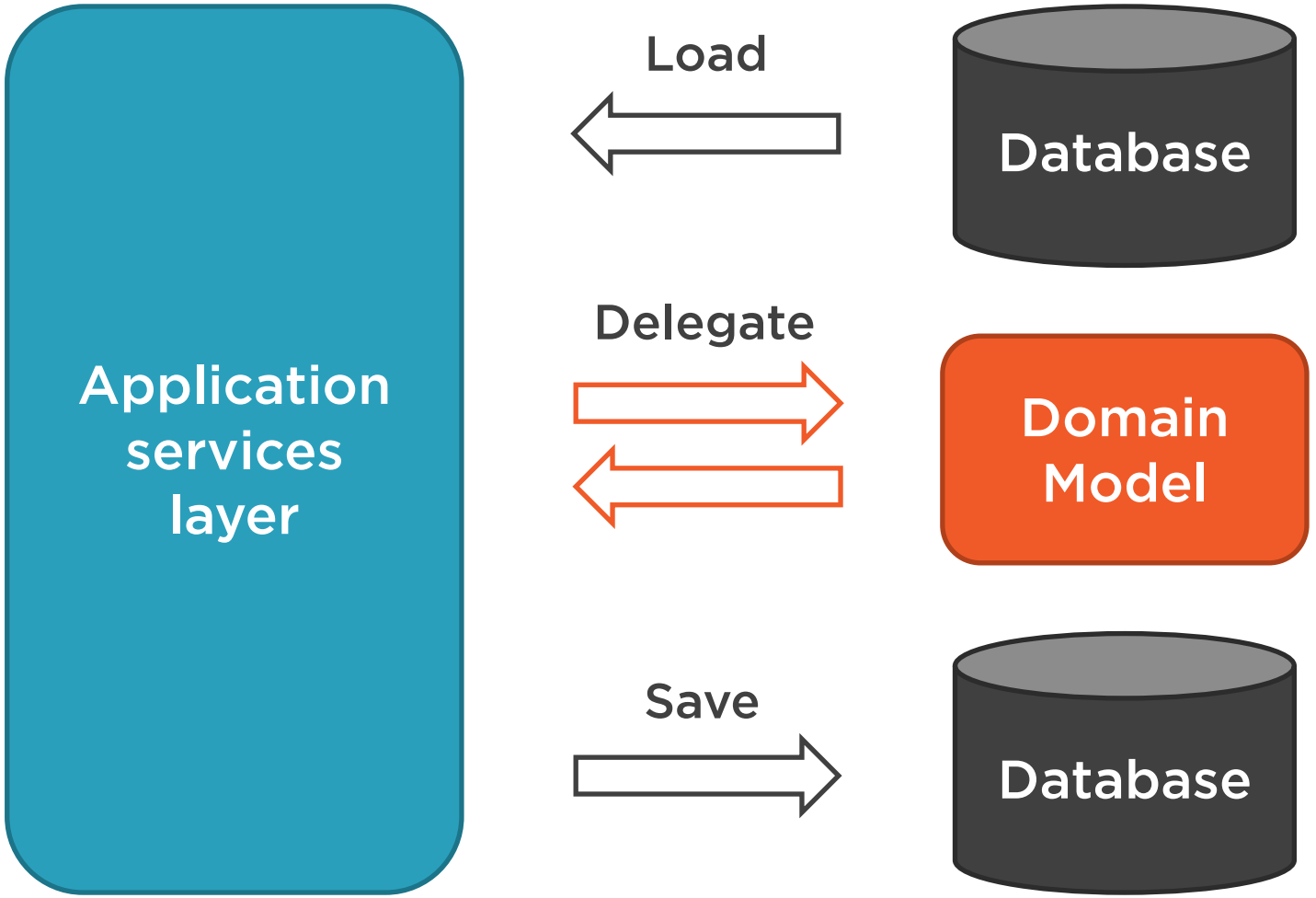
Protecting invariants

Domain Model Isolation

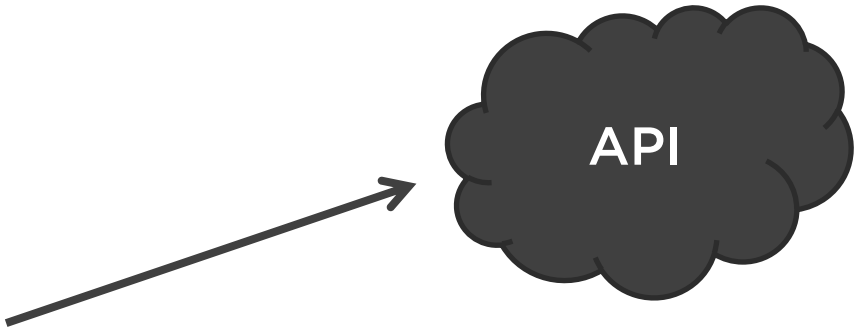


- ✓ Active record pattern
- ✓ Works on small projects
- ✗ Doesn't scale

Domain Model Isolation



Injecting Impure Dependencies into Domain Classes




```
public class Order
{
    public Price CalculatePrice(CurrencyConverter converter)
    {
        /* ... */
    }
}
```

```
public class Order
{
    public Price CalculatePrice(ConversionRates rates)
    {
        /* ... */
    }
}
```

Pure

Domain Model Isolation

Why isolate the domain model?

- Reducing complexity
- Increasing testability
- “Building a Pragmatic Unit Test Suite” course

Module Summary

Best practices and common anti-patterns in Domain-driven design

YAGNI principle

Don’t share domain logic between bounded contexts

Domain model encapsulation

Domain model isolation

Resource List

Source code	https://github.com/vkhorikov/AnemicDomainModel
	http://bit.ly/anemic-code
Domain-Driven Design in Practice	https://www.pluralsight.com/courses/domain-driven-design-in-practice
	http://bit.ly/ddd-ps
Domain model isolation	http://enterprisecraftsmanship.com/2016/09/01/domain-model-isolation/
	http://bit.ly/isol-1
How to know if your domain model is properly isolated?	http://enterprisecraftsmanship.com/2016/10/05/how-to-know-if-your-domain-model-is-properly-isolated/
	http://bit.ly/isol-2

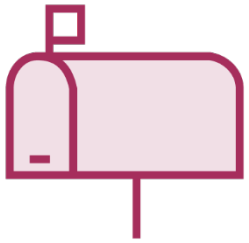
Course Summary

Anemic domain model: why it is an anti-pattern and how to refactor away from it

- Extracted explicit data contracts
- Used value objects as building blocks
- Pushed the domain logic from services down to domain entities and value objects
- Organized the application services layer

Domain modeling best practices and anti-patterns

Contacts



http://bit.ly/ddd-new

vladimir.khorikov@gmail.com

@vkhorikov

http://enterprisecraftsmanship.com/