Validating Data Across Client-Server Borders

Kevin Dockx
ARCHITECT

@KevinDockx www.kevindockx.com

Coming Up

The Principles of Validation

Focus on

- Where to validate
- How to structure our code

Dealing with Validation

Defining validation rules

Checking validation rules

Reporting validation errors

Defining and Checking Validation Rules

With Angular reactive forms, we can define validation rules in the class

Reactive forms make defining complex rules easier

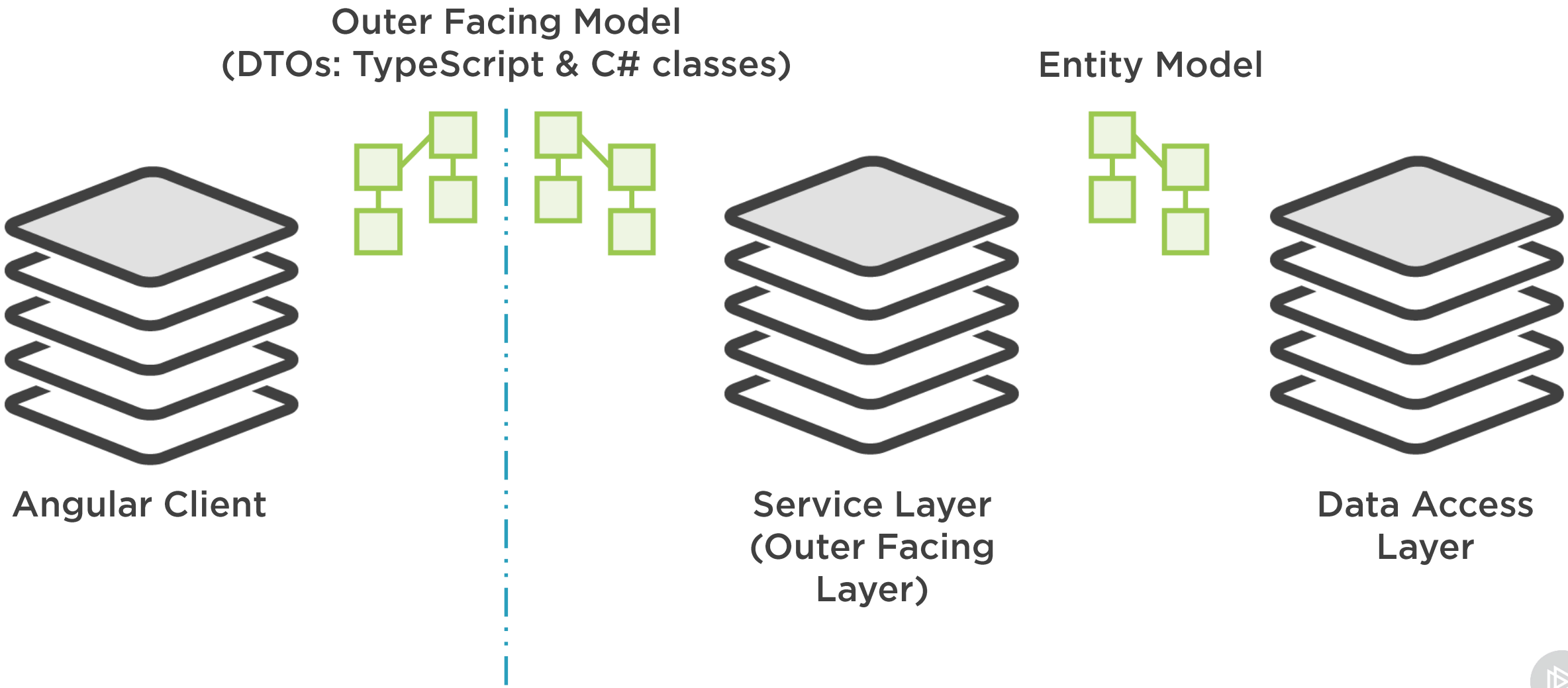
Defining and Checking Validation Rules

In ASP.NET Core rules can be defined with data annotations and through IValidatableObject

Stability and Data Integrity

Improve these by validating specific rules at the correct level

Dealing with Validation



Dealing with Validation

Client

- Catch validation errors at the earliest possible moment
- Be as strict as possible

... don't trust client-side validation

... don't trust data that crosses client-server borders

Dealing with Validation

When reaching the server

- Implement at least the same rules as at client level
- Additional rules are often implemented

Dealing with Validation

Validate at each layer

- Each layer works on its own model, one that's conceptually and technically different from that of another layer
- When data travels through layers, it's mapped to different types of objects with potentially different validation rules

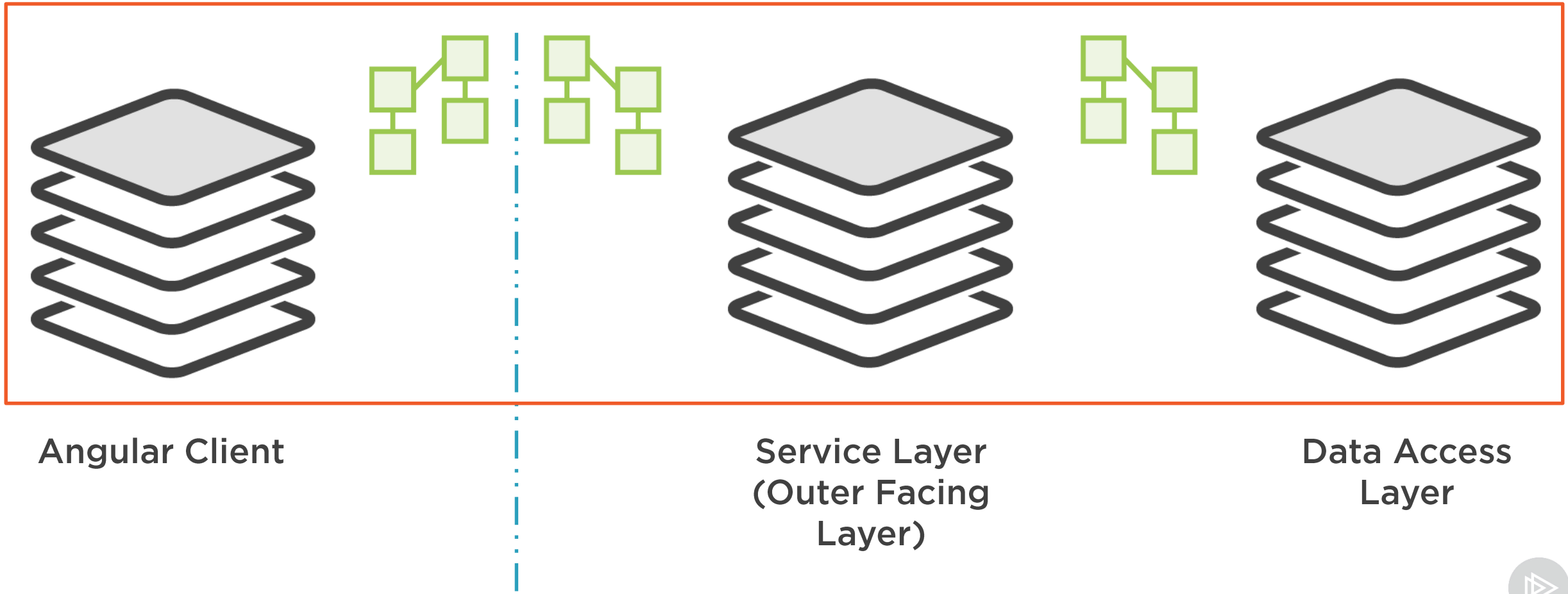
Dealing with Validation

When possible, incorporate lower-level rules in upper-level layers

- Capture errors early to avoid invalid objects trickling down
- … but still check the rules on each layer as code can be called from multiple places

Supporting Property-level Validation

Rule #1: title is a required field



Demo

Supporting Property-level Validation (Client)

Demo

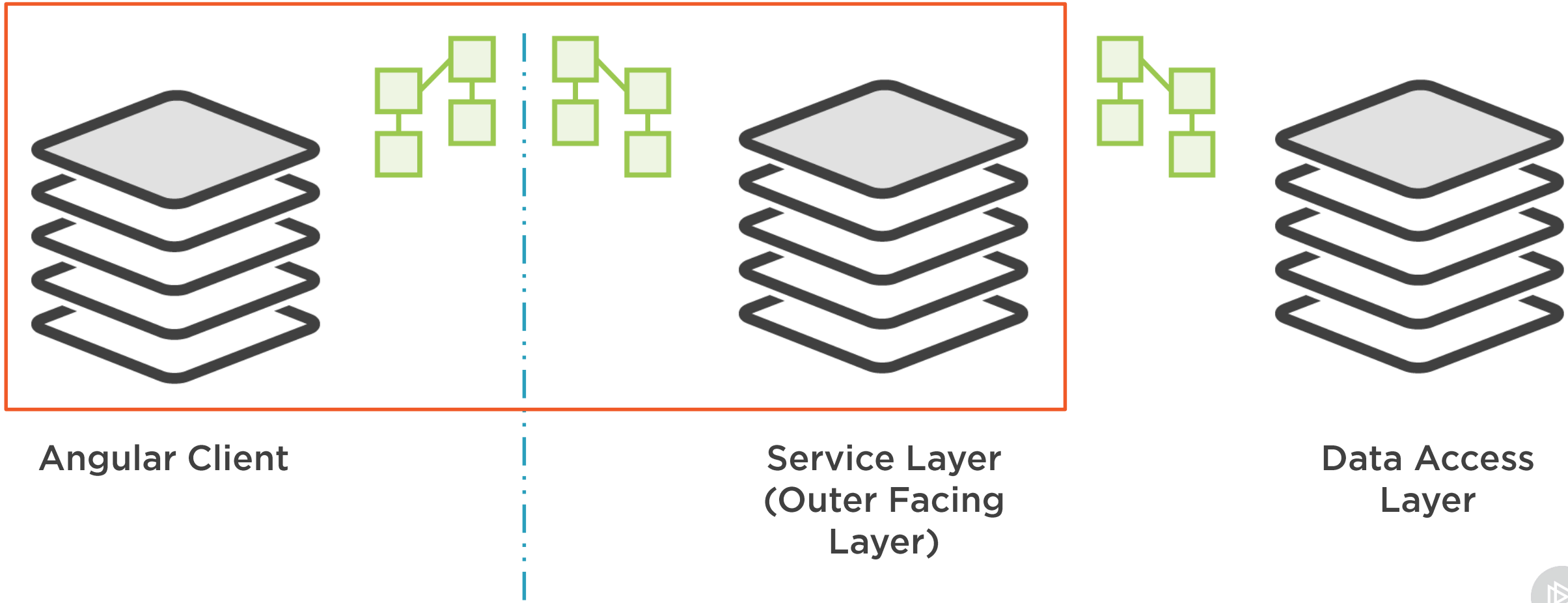
Supporting Property-level Validation when Creating a Resource (Server)

Demo

Supporting Property-level Validation when Updating a Resource (Server)

Supporting Object-level Validation

Rule #2: the start date must be smaller than the end date



Demo

Supporting Object-level Validation (Client)

Demo

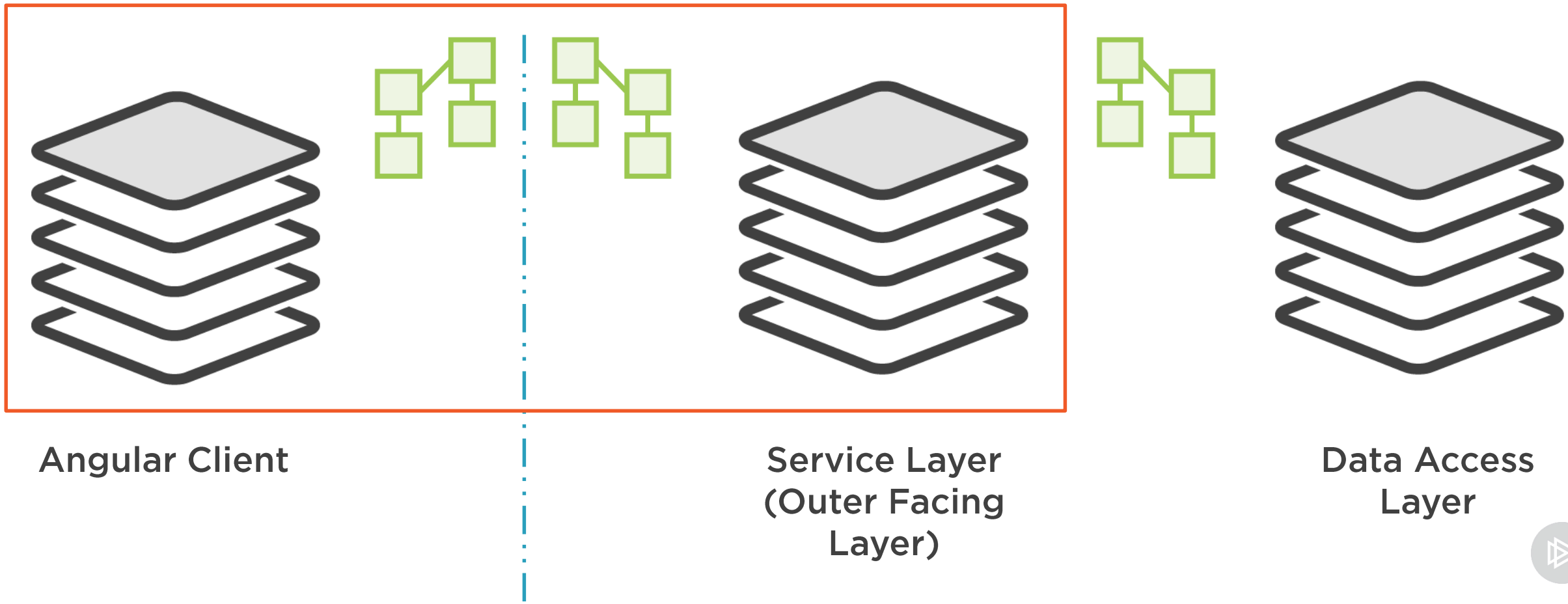
Supporting Object-level Validation (Server)

Demo

Supporting Object-level Validation with IValidatableObject (Server)

Handling Different Rules Between Creating and Updating Resources

Rule #3: when a tour is updated, the description is required



Demo

Handling Different Rules Between Creating and Updating Resources (Client)

Demo

Handling Different Rules Between Creating and Updating Resources (Server)

Summary

Each layer can have different validation rules

Errors should be caught as soon as soon as possible

We should be strict

Summary

Don’t trust client-side validation nor data that crosses client-server borders

- Client-side rules don’t help with data integrity, they help improve the user experience

At server level, catch errors as soon as possible

- Incorporate lower-level rules in upper-level layers, and add to them

Summary

Work with different models and separate classes out according to their responsibility

- Improves data integrity
- Allows for use cases that would otherwise be impossible