Denver Dev Day
November 22nd, 2019

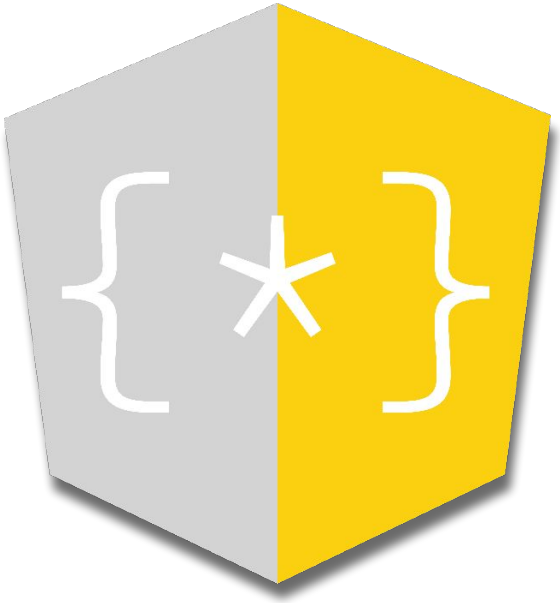
Join our MeetUp:
http://bit.ly/DDD-mu

Thanks to our Sponsors!

Microsoft

Signup for Microsoft Source: aka.ms/usdevconnect

revgen



Angular Architecture

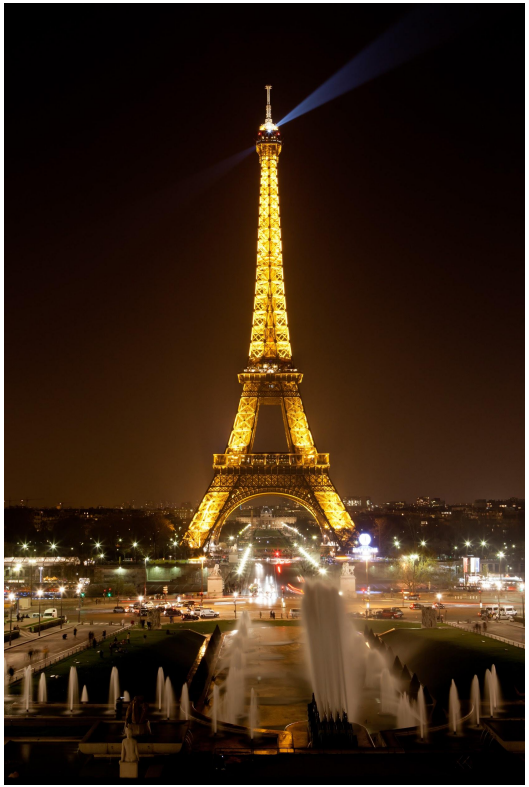
Take Angular Architecture to the CLEANers

Matt Vaughn
@angularlicious
matt@angularlicio.us

“

What is architecture?

Architecture Examples



EFFECTIVE ARCHITECTURE

- Experience
  - Knowledge, understanding
  - Analysis, design, planning
  - Technical Leadership
- Essentials
  - Tools and materials
- Execution
  - Plan, recipes
  - Guard the code

“

Why does architecture matter?

CLEAN ARCHITECTURE

Use CLEAN code and architectural approaches.

CLEAN CODE

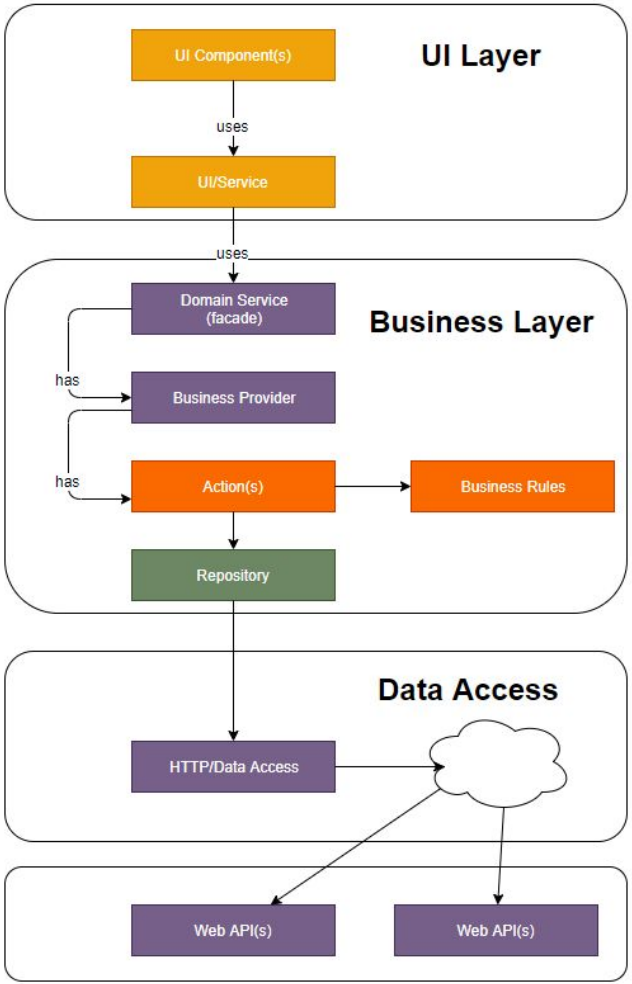
- Separation of Concerns
- Single Responsibility
- Clear boundaries between layers
- DRY Code
- Layered Architecture
- Design Patterns
- Object Oriented Practices

CLEAN ARCHITECTURE

- Layers
- Boundaries
- Inversion of Control (DI)
- *Testable*
- *Independent (frameworks, UI, database)*

Architecture HISTORY

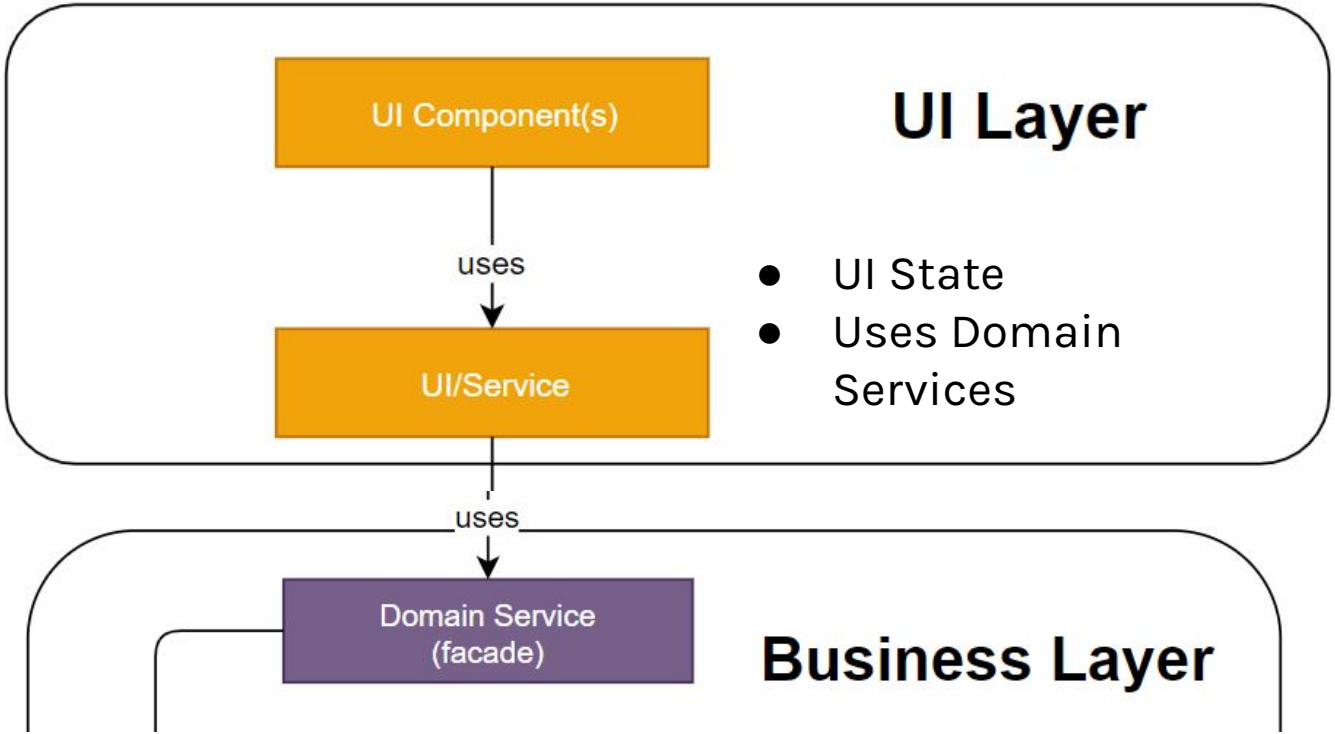
- Layers (n-tier)
- Hexagonal
- Clean
- Onion


UI Component(s)
UI Layer
uses
UI/Service
uses
Domain Service (facade)
Business Layer
has
Business Provider
has
Action(s)
Business Rules
Repository
Data Access
HTTP/Data Access
Web API(s)
Web API(s)

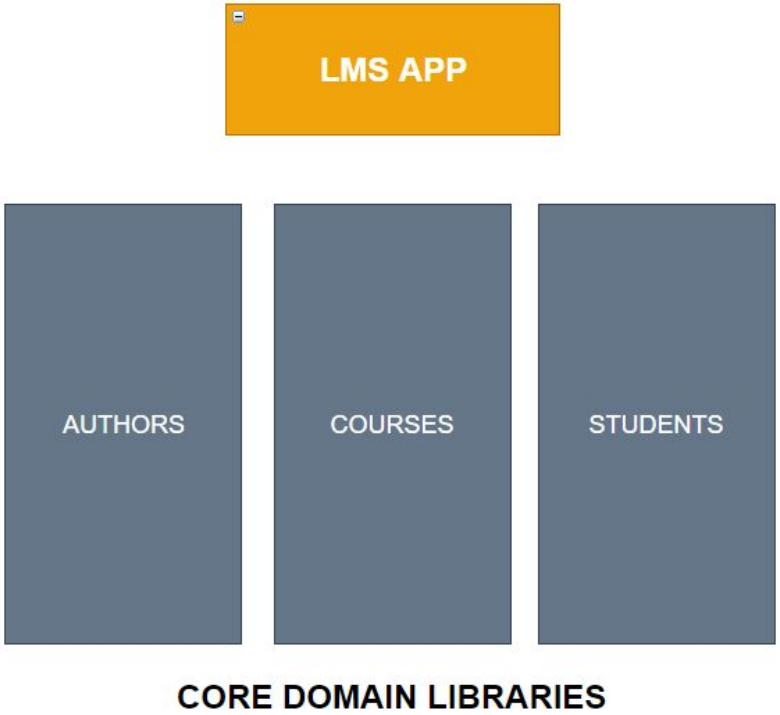
LAYERS SEPARATE CONCERNS

- UI Concerns
- Domain Services
- Business Logic Concerns
  - Actions
  - Rules
- Data Repository Concern
- Data Provider Concern/HTTP

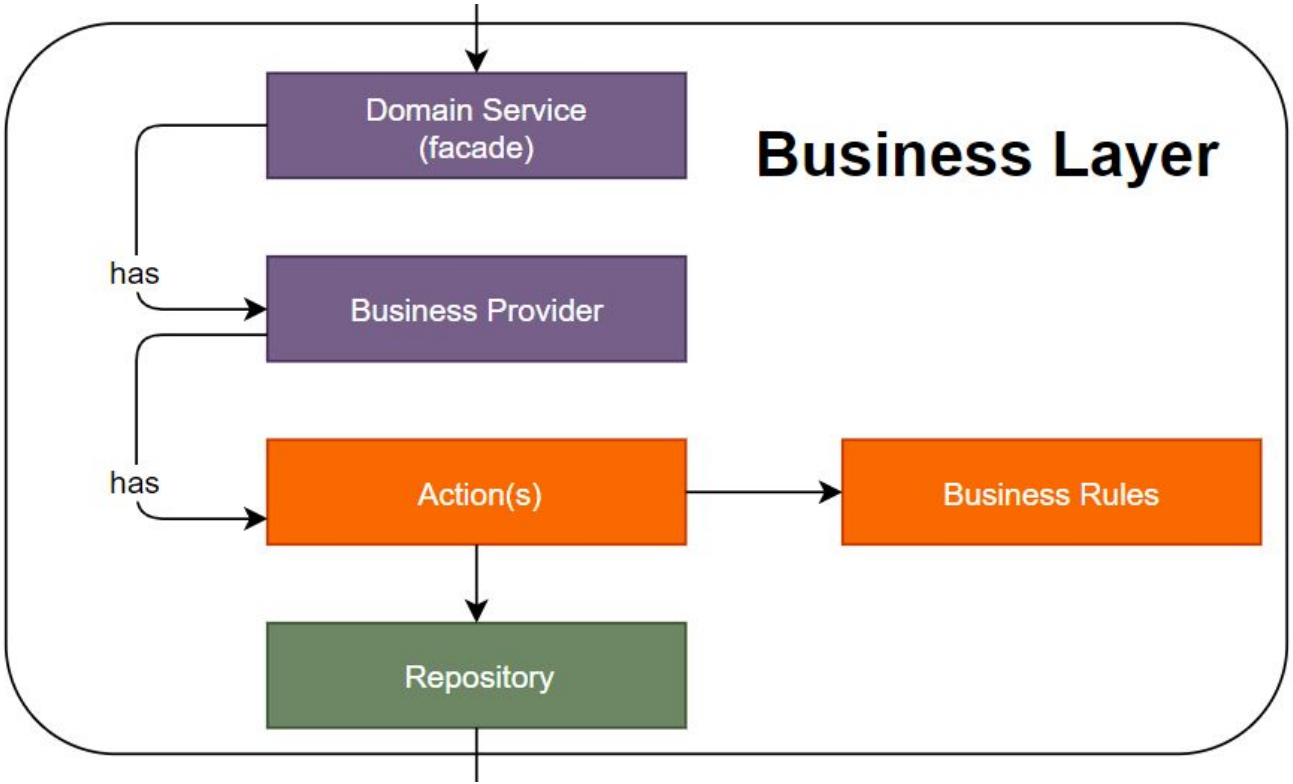
UI CONCERNS


UI Component(s)
uses
UI/Service
UI Layer
UI State
Uses Domain Services
uses
Domain Service (facade)
Business Layer

DOMAIN SERVICE CONCERNS



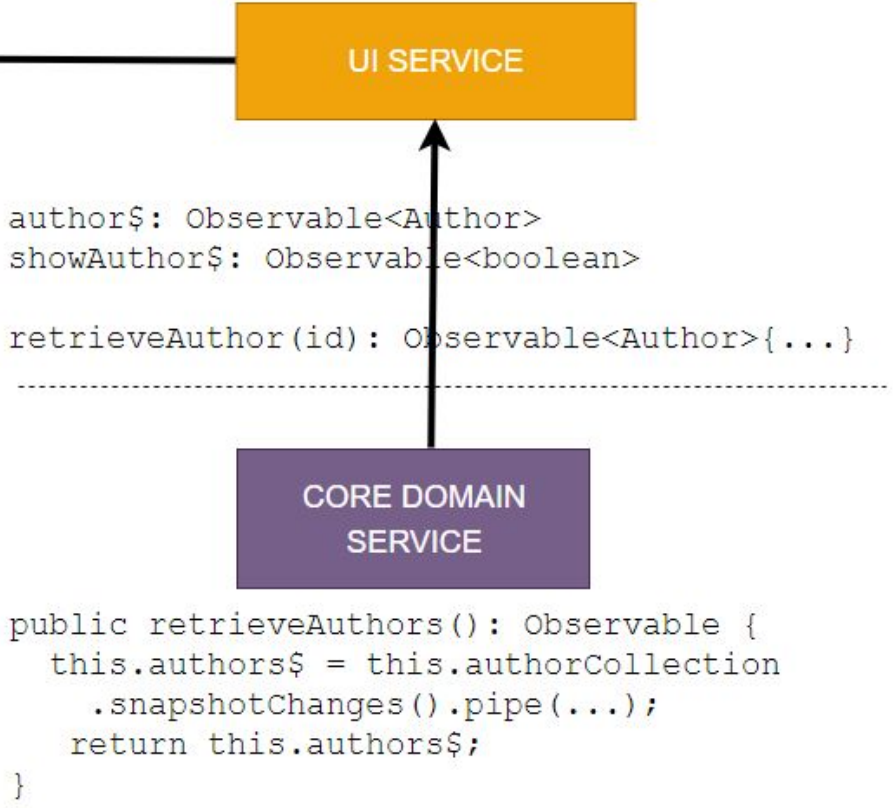
DOMAIN SERVICE CONCERNS



ONE-WAY DATA

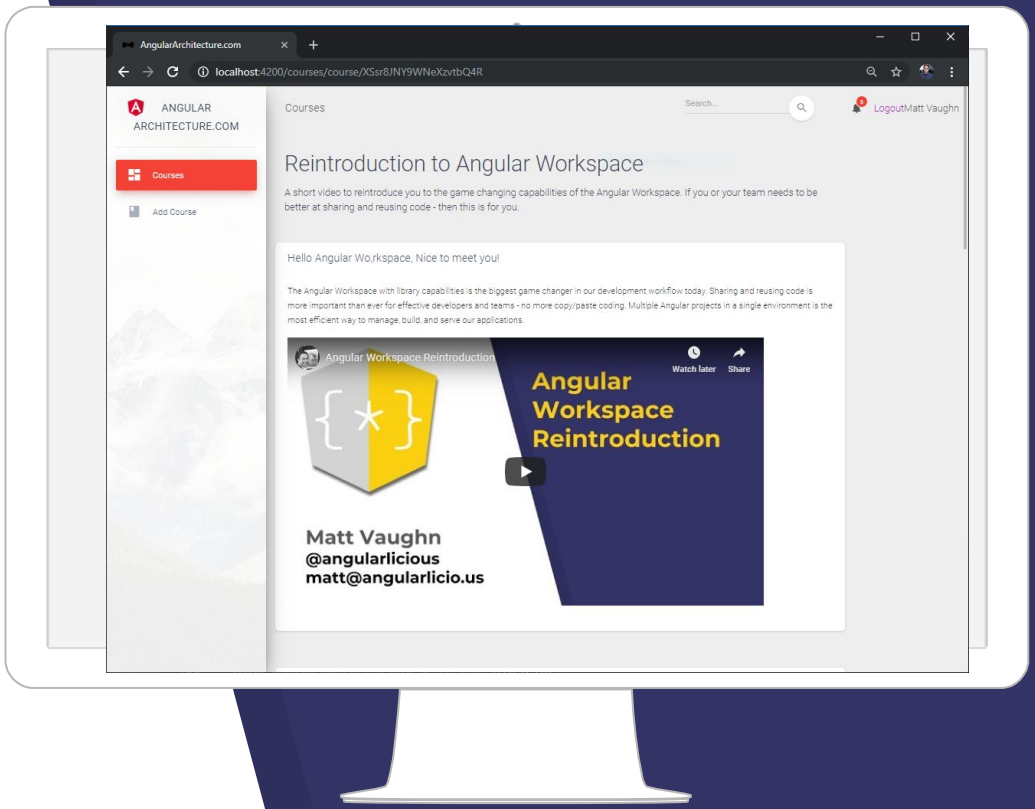

CONTAINER COMPONENT
*ngIf="showAuthor$ | async"
<app-cmpnt-two [author]="author$ | async"
PRESENTATION COMPONENT
@Input() author: Author
<h3>Author information:</h3>

{{ author.bio }}

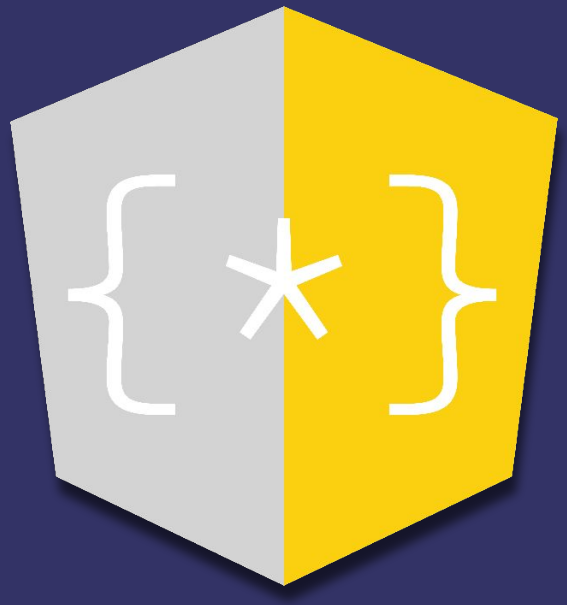

UI SERVICE
author$: Observable<Author>
showAuthor$: Observable<boolean>
retrieveAuthor(id): Observable<Author>{...}
CORE DOMAIN SERVICE
public retrieveAuthors(): Observable {
this.authors$ = this.authorCollection
.snapshotChanges().pipe(...);
return this.authors$;
}

CODE REVIEW

Let’s review a workspace with an application and several library projects.



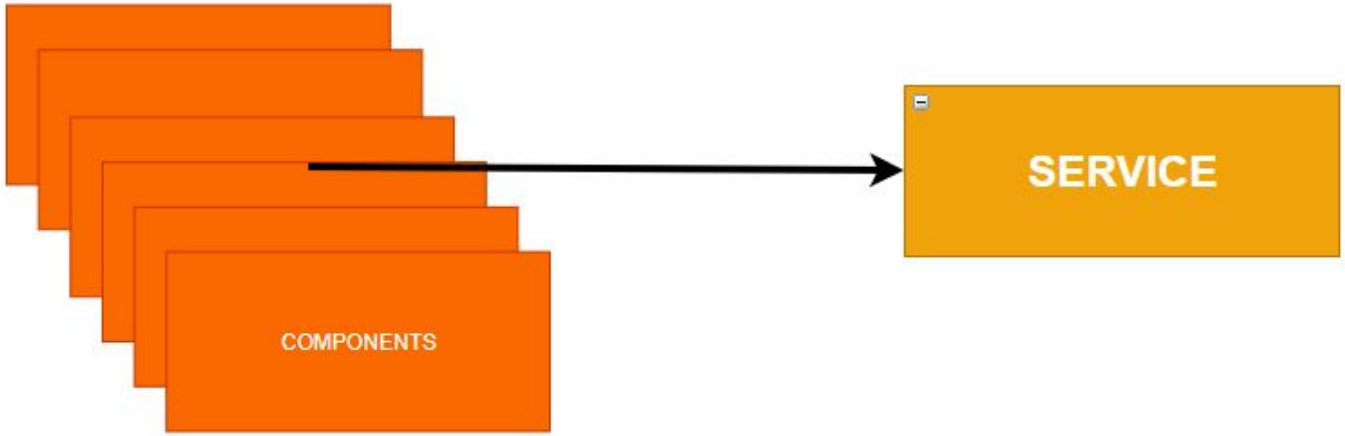
Using Libraries



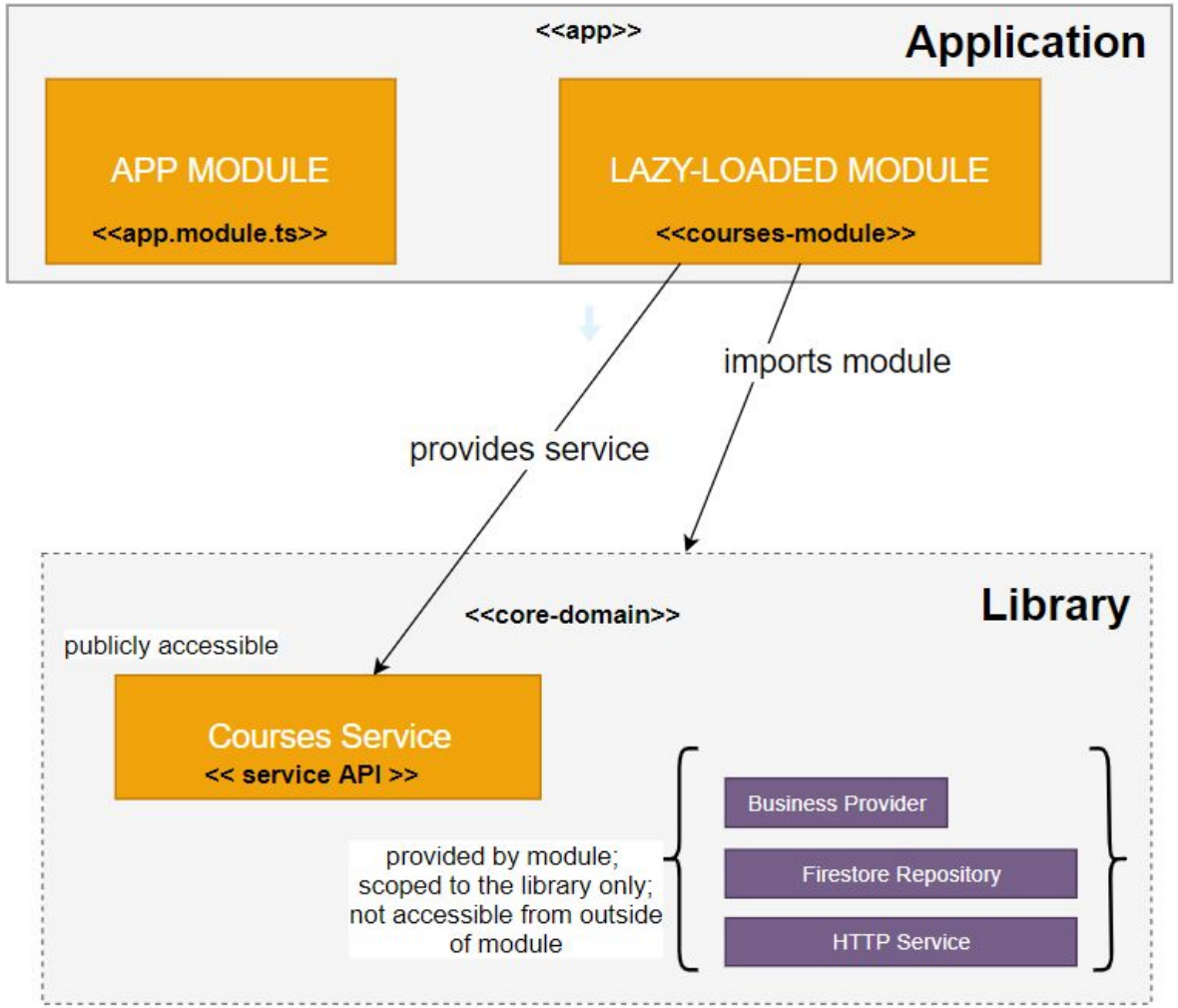
Is your Service OVERWORKED?

SERVICE OVERLOAD

CODE...



Many services are loaded down because they have to manage state, provide API-like methods, perform business logic, validate input, process business rules, make HTTP requests, handle responses, handle HTTP errors, log events, etc.


<<app>>
Application
APP MODULE
<<app.module.ts>>
LAZY-LOADED MODULE
<<courses-module>>
imports module
provides service
<<core-domain>>
Library
publicly accessible
Courses Service
<< service API >>
provided by module; scoped to the library only; not accessible from outside of module
Business Provider
Firestore Repository
HTTP Service

CODE...

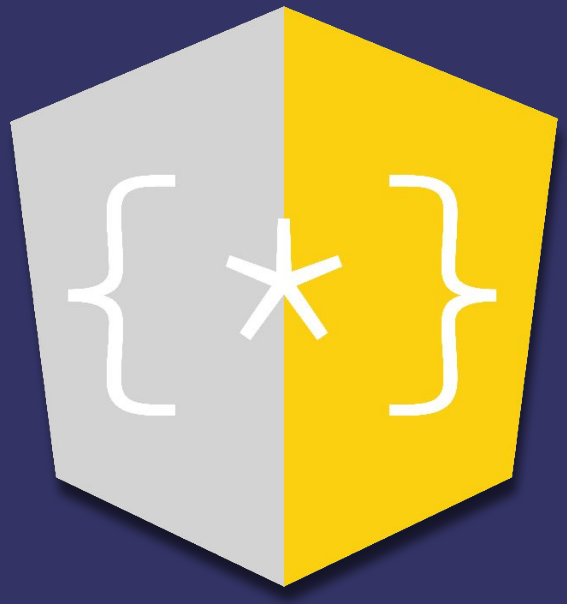
CUSTOM LIBRARIES

- Single Library Projects
- Angular Package Format
- Publishing and Consuming
- Versioning
- Not an efficient developer workflow

WHY LIBRARIES?

- Enables code reuse
- Maintenance
- Code Organization
- Publish and share

Library Types



DIFFERENT LIBRARY PROJECT TYPES

Cross-Cutting Concerns

Use to provide concerns that cross the boundaries of the application/domain: logging, error handling, configuration, etc.

Foundational

Use Object-Oriented programming to provide base classes for Angular Services, Components, Business Actions.

Components

A library for common components, pipes, and directives.

Feature Library

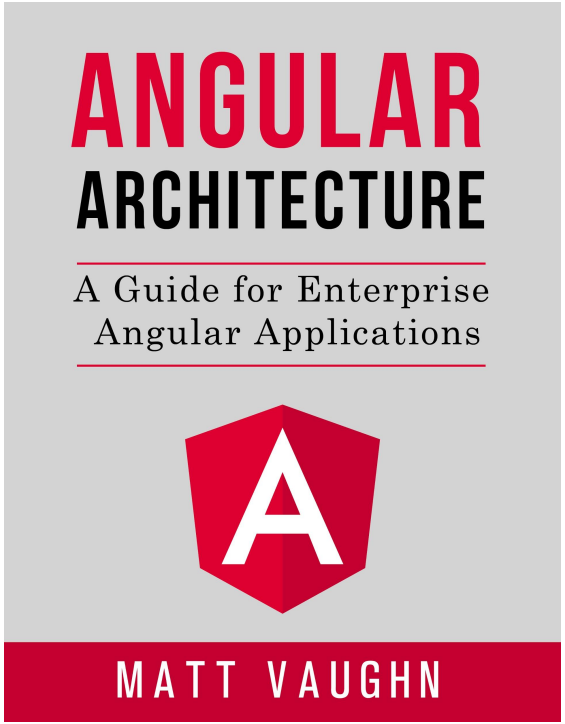
Use a library to encapsulate a feature library that can be used by multiple apps (security, shopping cart).

Core/Domain Library (Business Logic Layer)

Use a library to implement rich business logic layers that are accessed by service facades.

Framework

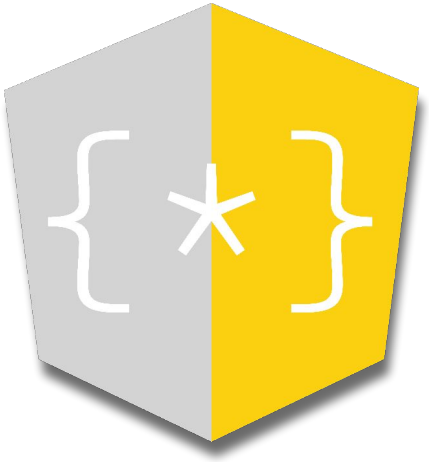
Create reusable frameworks to provide consistent implementation: rule engine, validators, business actions.



Free book for Denver Dev Day at:

BOOK: http://bit.ly/DDD-ANGULAR-ARCHITECTURE

Web: www.leanpub.com



@angularlicious

Email: matt@angularlicio.us

Web: www.angularlicio.us

Blog: www.medium.com/@angularlicious

Github: www.github/angularlicious

RESOURCES

All of the resources for this mini-workshop are available at:

- GitHub/**angularlicious**
- **Repo:**
  - **angular-connect-workspace**