Domain Model For Angular Apps

What’s The Domain?

The domain is the heart of the application

- How would you describe the purpose of the app?
- The critical objects that the app manipulates (devoid of external technologies – UI, database, security)
- Sometimes called the business layer or the logic layer

Domain’s Stability

Over a ten-year lifespan of a large system, domain will be stable (though will evolve)

- The domain defines what the app does

Everything else will likely change

- Database technologies,
- messaging APIs and
- User interface frameworks

Sensible to invest time to get domain model right

In the past ...

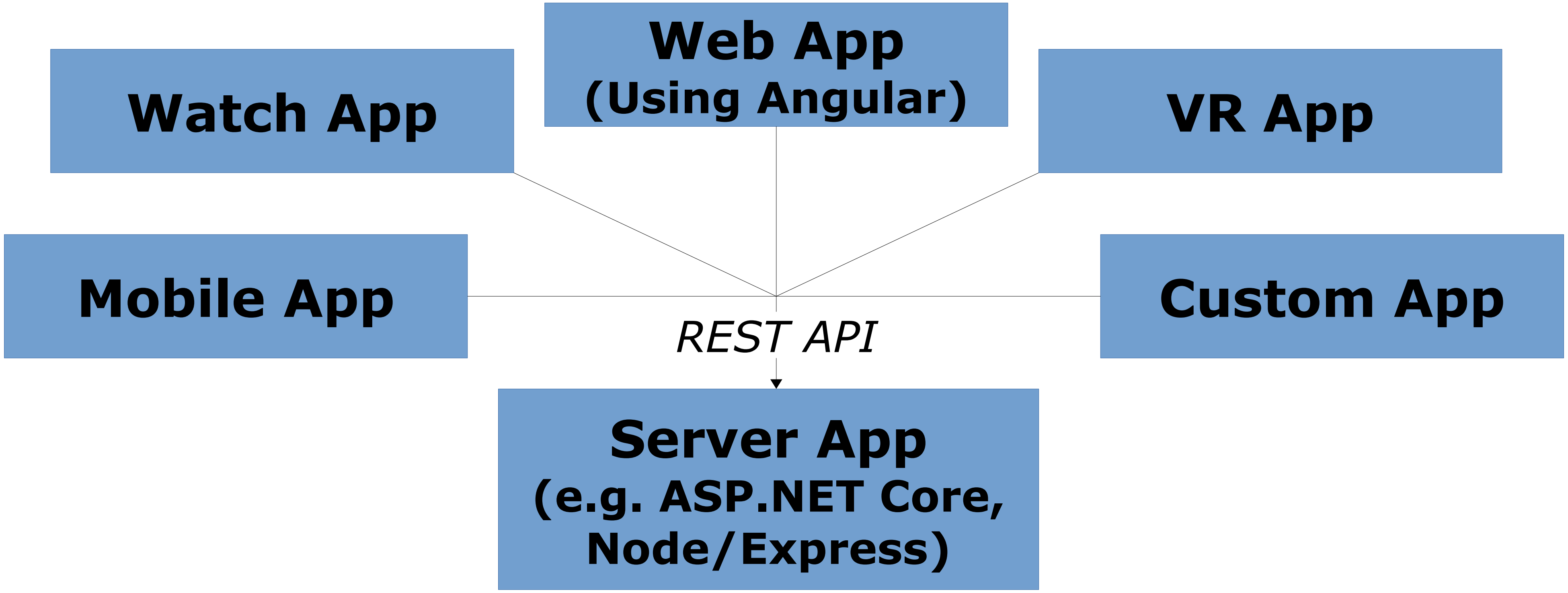
App ran on server

- MVC on the server

What was sent to web browser was just output and simple data returned

- Small amount of JavaScript code
- All the main logic ran on server

Now..many apps..each evolves at own pace



Role of REST API

Exchange of messages between client and server

- Usually JSON based
- Be aware of latency issues
- Cost of data transfers

Angular offer an HTTP library that should be used

- Allows in-memory mocking of HTTP client
- Responses returned as observable event emitters
- XSRF Strategy

API Gateways

API could travel through API gateway

- Client and server evolve at different pace
- e.g. app for phone can take time to get into app store, whereas a web app can be immediately updated
- So new and old APIs may have to be handled for a period of time

Microservice on Server

Bounded contexts in an Angular app might map very nicely to distinct microservices on the server

- Maybe not one-to-one
- There is correlation among some features

What’s The Application?

With Angular, what runs in web browser is much more sophisticated

Makes sense to treat it as distinct app

- e.g. URL that appears in browser’s address bar can be manipulated by browser app (and with Angular’s router, this is exactly what happens)

MVC For Angular Apps

MVC moves to the web browser

- Web applications getting larger, an app in their own right
- What happens on server is separate world (separate app), accessed via REST API

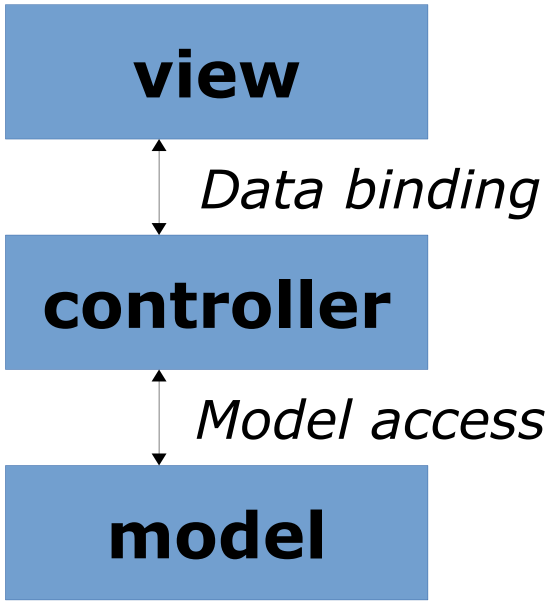
View

- In Angular world, this is the template

Controller

- In Angular world, this is the component

Review of MVC



Model

- Domain model (not data model)

View

- What appears on screen

Controller

- Reads from model, manipulates its data, and data binds to view; and also reverse

Model Driven Design

Application Model

Interaction Model

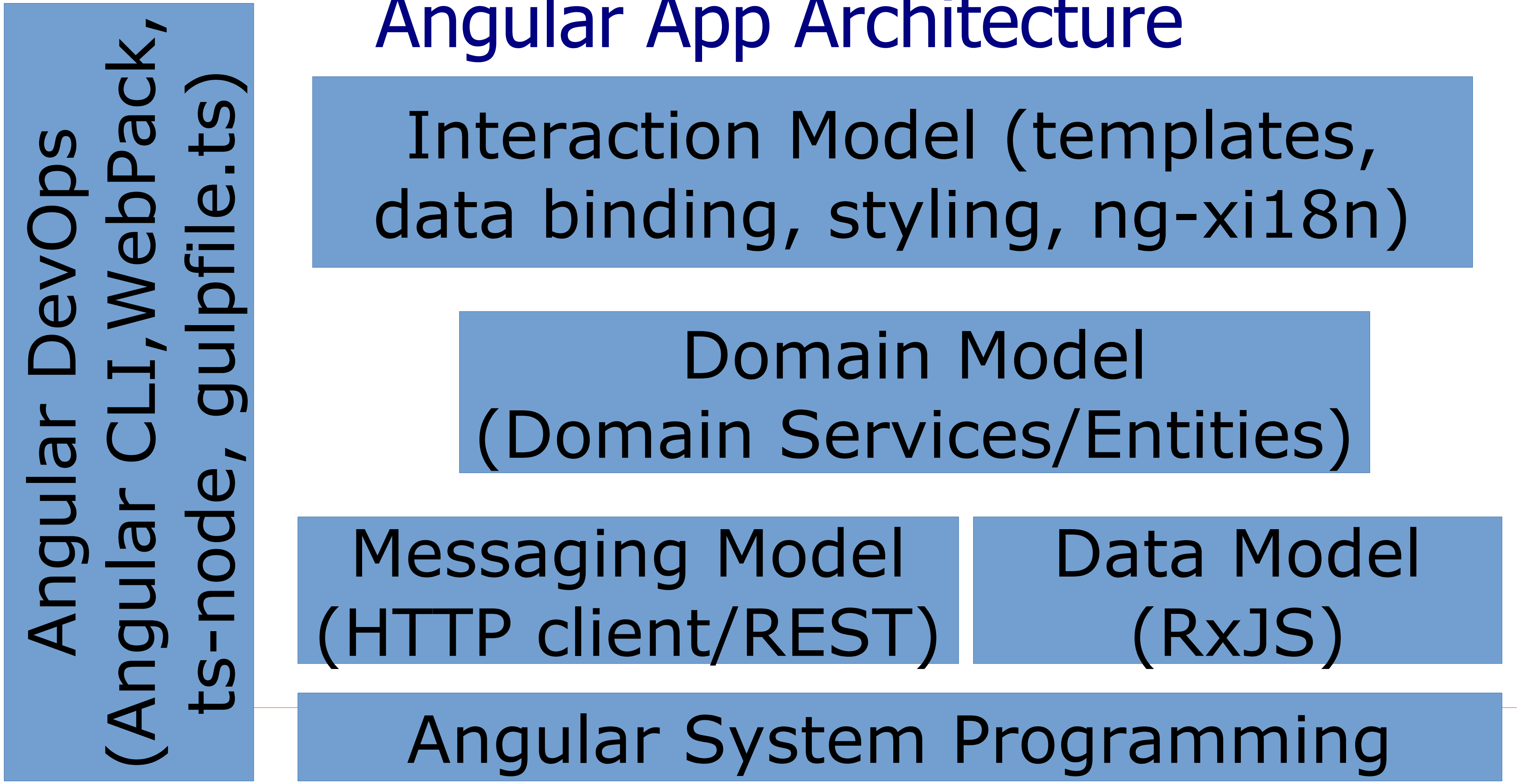
Domain Model

Messaging Model

Data Model

- Lots more … packaging model, security model, build model, configuration model

Angular App Architecture


Interaction Model (templates, data binding, styling, ng-xi18n)
Domain Model (Domain Services/Entities)
Messaging Model (HTTP client/REST)
Data Model (RxJS)
Angular System Programming
Angular DevOps (Angular CLI,WebPack, ts-node, gulpfile.ts)

Domain Constructs (1)

Entity

Value Object

Aggregate

Domain Constructs (2)

Repository

Service

Factory

Domain Services

All services are not domain services

- Think logging, performance monitoring

But most are

- E.g. PurchaseOrders.getList

Role of RxJS

Ubiquitous Language

Language And Software development

- Words are important

In any business domain, there is terminology and specific meanings

- Best to leverage such ubiquitous language as the project language
- Talks with (non-tech, but highly knowledgeable) domain experts will be more fruitful

Intention Revealing Interfaces

The clients of a type are not really interested in the low-level implementation details

The interface

An interface should clearly expose the intention of the type

- Not its internals

Bounded Context

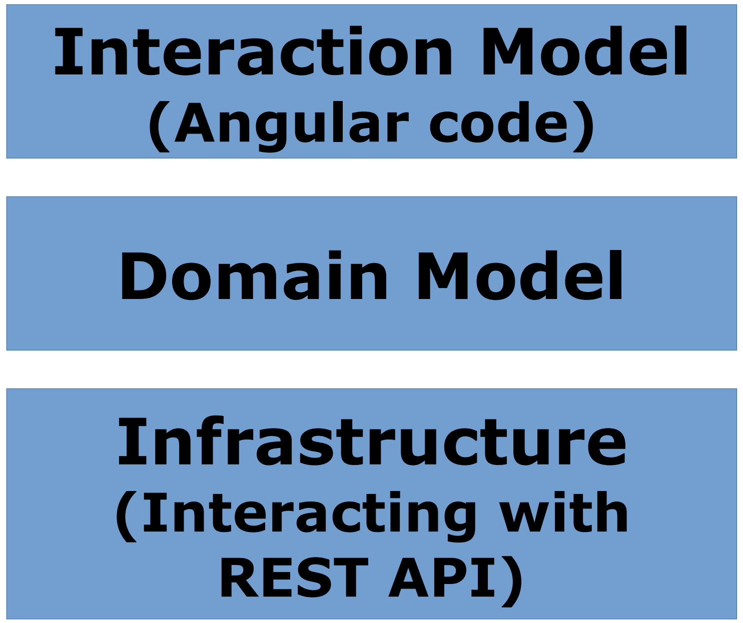
Within a domain model there are often distinct clusters of entities and related

- Sometimes it makes sense to more formally define these boundarties

Use Bounded Contexts

- Could go so far as to have separate packages for separated bounded contexts
- Model communication between bounded contexts

Application Layering


Interaction Model
(Angular code)
Domain Model
Infrastructure
(Interacting with
REST API)