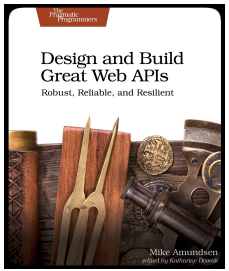
Designing Great APIs
Part One

@mamund
Mike Amundsen

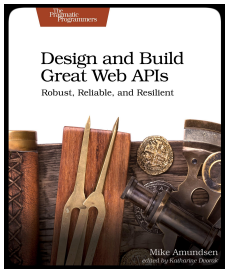


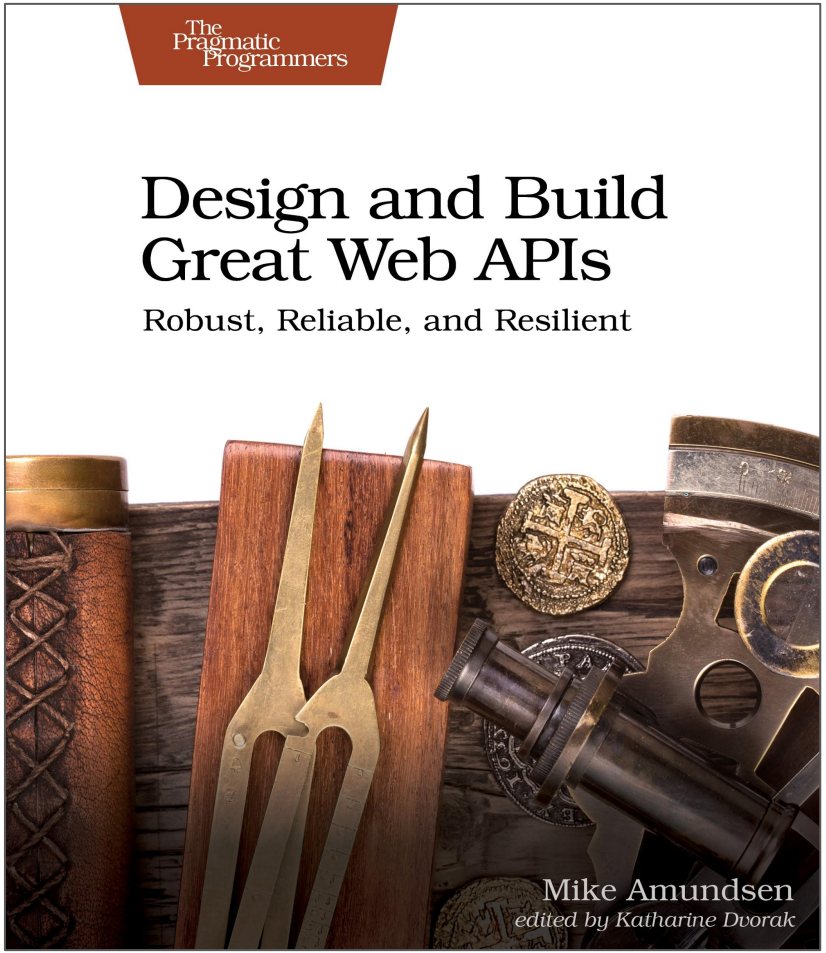

Mike Amundsen
@mamund

g.mamund.com/GreatWebAPIs

"From design to code to test to deployment, unlock hidden business value and release stable and scalable web APIs that meet customer needs and solve important business problems in a consistent and reliable manner."

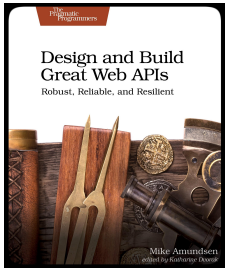
-- Pragmatic Publishers





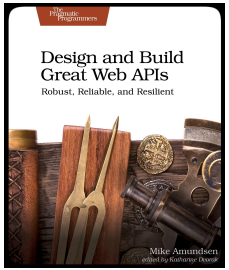
Logistics and Preparation

- Introductions
- Workshop Outline
- Zooming



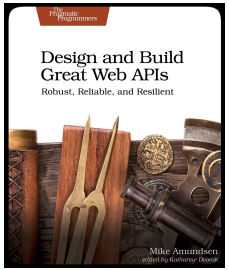
Introductions

- Name
- Current work
- What you're hoping to learn



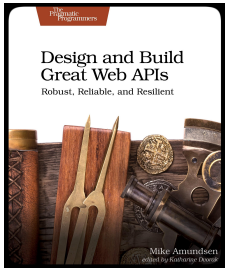
API Design Workshop

- Part One (today)
  - Stories, Models & Diagrams
  - Design Method
  - Overnight Assignment
- Part Two (tomorrow)
  - Assignment Review
  - API Descriptions

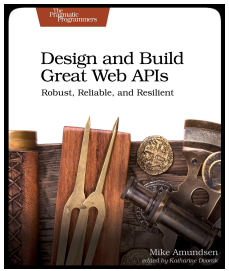


Zooming

- Share video feed on whenever possible
- Mute your microphone when not talking
- Raise your hand to share, ask questions, etc.
- Add background questions/comments in chat window
- If you need to leave your desk, turn video off

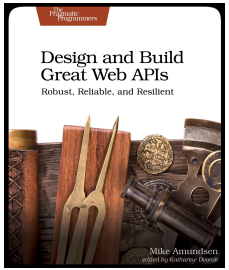


API Stories, Models and Diagrams



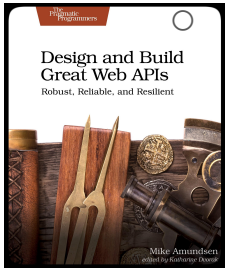
API Stories, Models and Diagrams

- The Importance of Stories
- The Power of Models
- The Value of Diagrams



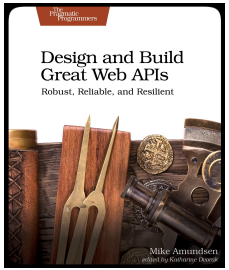
API Stories

- APIs start with a story
  - "We need…"
  - "Our customers requested…"
  - "I have an idea…"
- Stories are shared understanding
  - Our brains are wired for stories, not data
  - Stories are accessible
  - Stories are repeatable



API Story Builder

- Purpose
- Data
- Actions
- Processing
- Rules



23 lines (15 sloc) | 1.01 KB

Raw Blame History

Simple ToDo

Purpose

We need to track 'ToDo' records in order to improve both timeliness and accuracy of customer follow-up activity.

Data

In this first pass at the application, we need to keep track of the following data properties:

- **id** : a globally unique value for each ToDo record
- **body** : the text content of the ToDo record

Actions

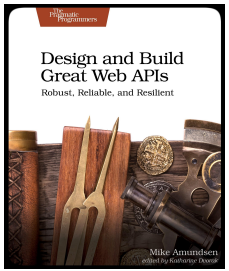
This edition of the application needs to support the following operations:

- **List** : return a list of all active ToDo records in the system
- **Add** : add a new ToDo record to the system
- **Remove** : remove a completed ToDo record from the system

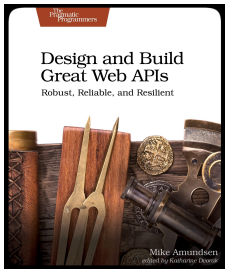
NOTE: In this edition, we will not track any history of completed ToDo items.

Rules

When creating a record, the client SHOULD generate and send a value for the **id** property. If the client does not send a value for the **id**, the service MUST generate one for that record. In both cases, the service MUST ensure the **id** value is unqiue and reject any writes that would result in duplicate **id** values.

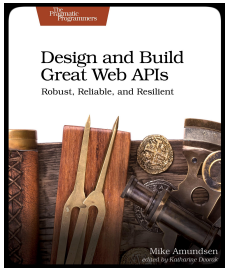


Every API starts with a story

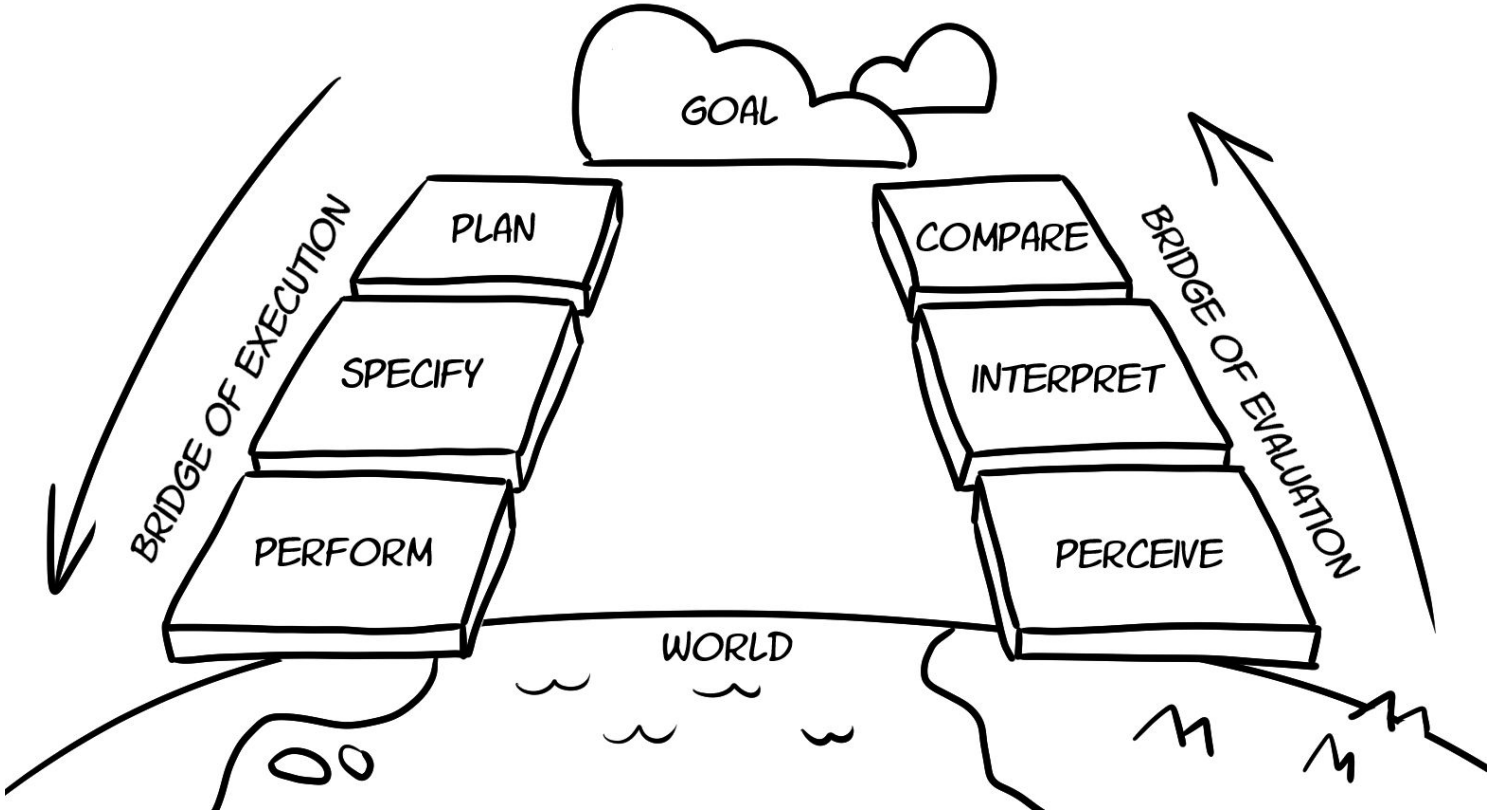


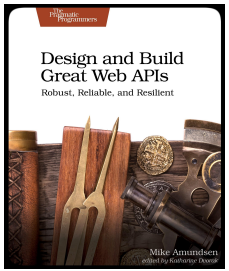
API Modeling

- Models are how we see the world
  - Donald Norman's Lifecycle
  - It's always a circle, not a line
  - The RPW Loop

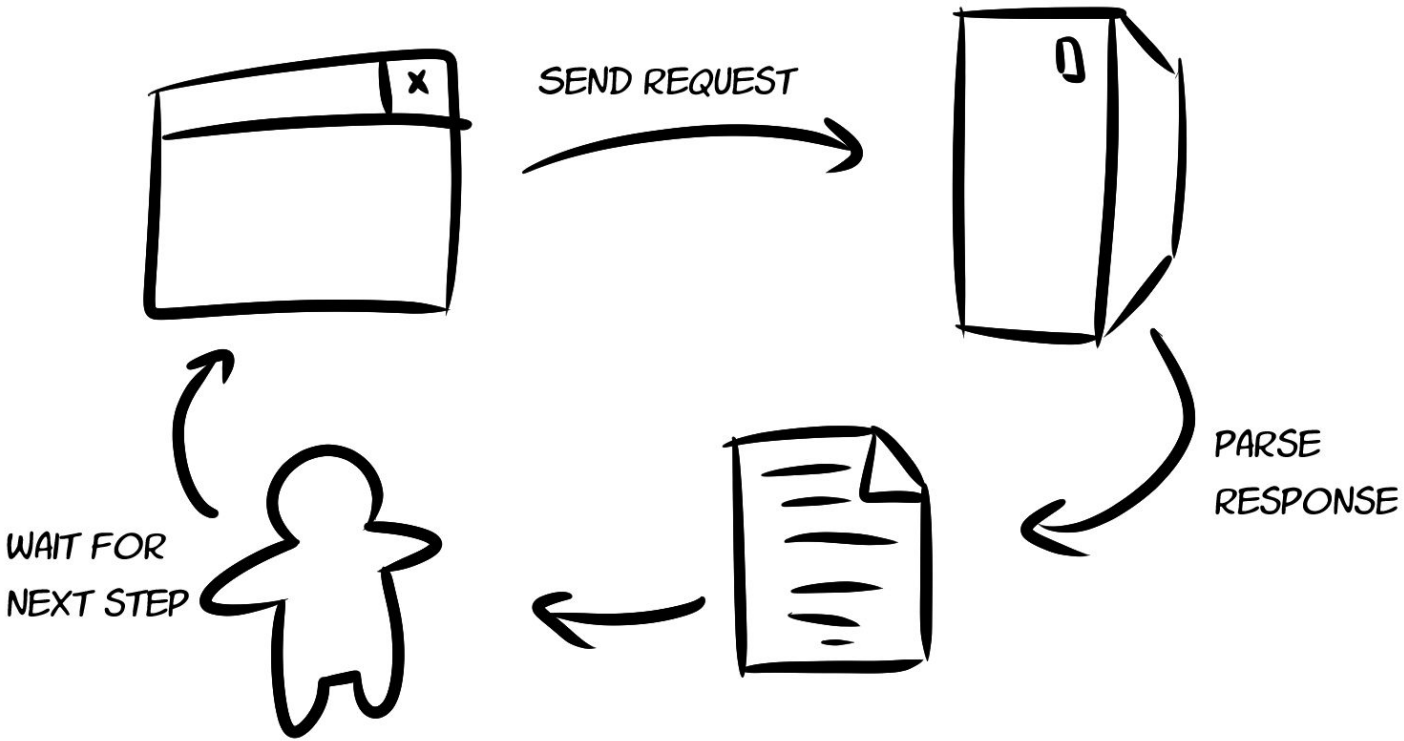


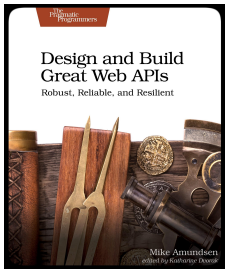
Norman's Lifecycle





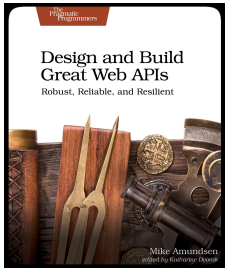
The RPW Loop





API Modeling

- Models are how we translate the story
  - Data
  - Actions
  - Workflows



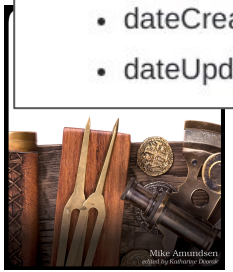
Company Vocabulary

Data Elements

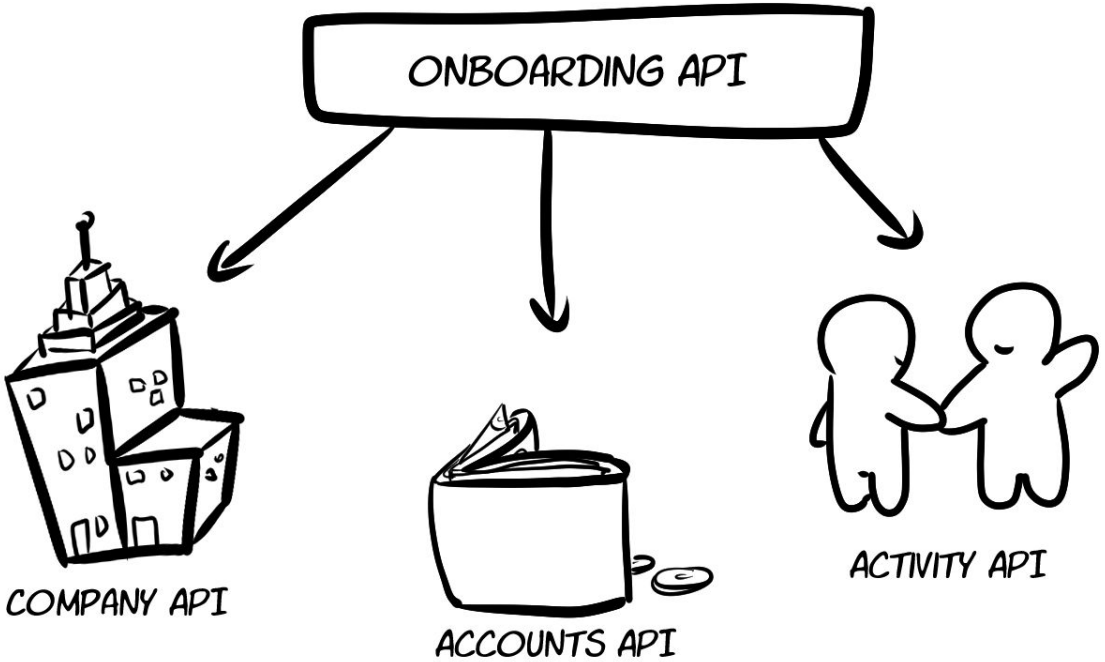
- companyId
- companyName
- streetAddress
- city
- stateProvince
- postalCode
- country
- telephone
- email
- status (suspended, active, pending,closed)
- dateCreated
- dateUpdated

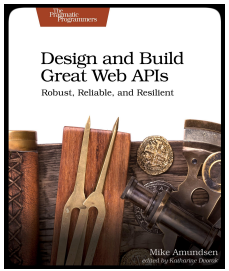
Action Elements

- list
- create
  - companyName[R], streetAddress, city, stateProvince, postalCode, country(US), telephone, email[R], status(pending)[R]
- read
  - companyId[R]
- update
  - companyId[R], companyName[R], streetAddress, city, stateProvince, postalCode, country(US), telephone, email[R], status[R]
- delete
  - companyId[R]
- filter
  - status, country, state/province, companyName

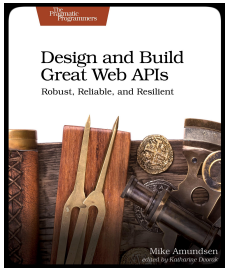


Workflow

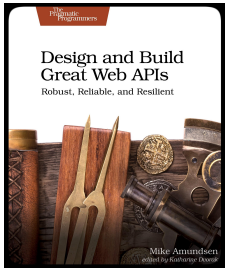




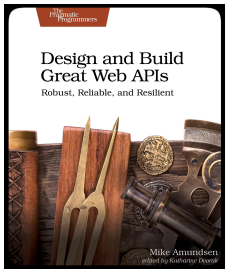
Models help us focus on a solution



Let's Discuss!

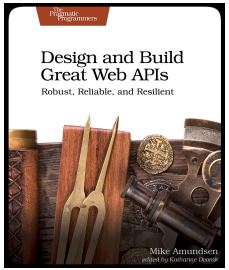


An API Design Method



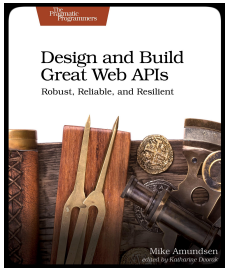
API Design Method

- API Design is not implementation
- Methods should be repeatable
- Methods should be teachable
- Methods should be trackable



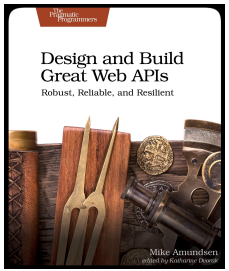
API Design Method

- Story
- Model
- **Normalize**
- Diagram
- Describe



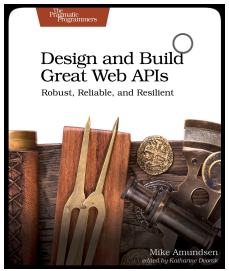
API Design Method

- Each API has its own vocabulary
- Properties
  - Firstname, lastname, zipcode, etc
- Actions
  - Save, Approve, Cancel, Share, etc.



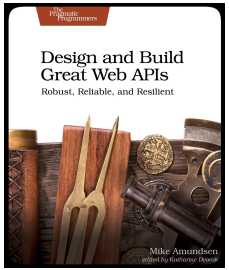
API Design Method

- APIs have their own "story" but share the same "terms"
- API Properties
  - firstname -> givenName
  - lastname -> familyName
  - zipcode -> postalCode
- Shared terms mean shared understanding
  - company.status = account.status
  - user.familyName = customer.familyName



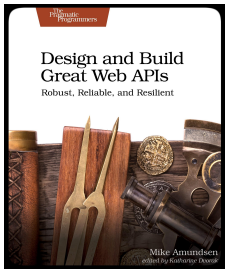
API Design Method

- Use shared dictionaries to normalize properties
- FIHR (for healthcare)
  - https://www.hl7.org/fhir/overview.html
- PSD2 (for banking)
  - https://en.wikipedia.org/wiki/Payment_Services_Directive

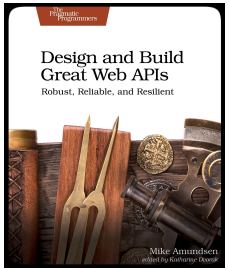


API Design Method

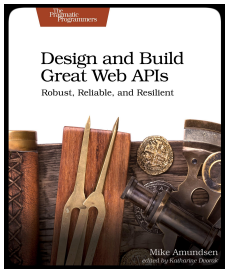
- schema.org
  - Generalized property dictionary
- Your own company
  - data models,
  - UML diagrams
  - Glossaries, etc.



Let's Discuss!

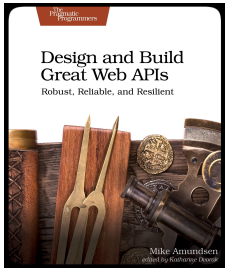


Normalizing Your API Design



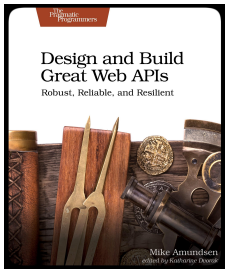
Normalizing the Onboarding API

- Review the API Model document
- Identify properties to normalize
- Identify vocabulary sources
  - Schema.org (for this exercise)
- Update Model to use schema.org terms

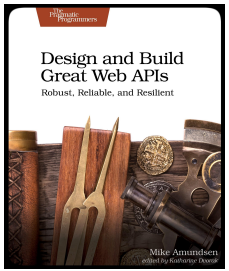


Normalizing the Onboarding API

- Focus on properties, not actions
- You may need to "compromise"
  - voicePhone -> telephone
  - Id -> identifier
- You may need to roll your own
  - Use URI syntax
  - http://api.mamund.com/terms/hatsize

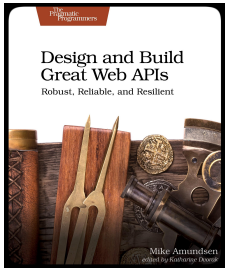


Exercise: Normalizing your API Design

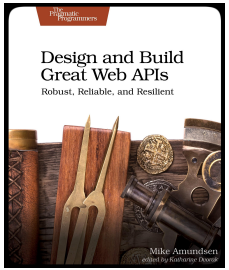


Normalizing your API Design

- Use the diagram/story you started with
- Be sure to write up your Model document
- Use schema.org as your dictionary
- Document changes/references in your API Model

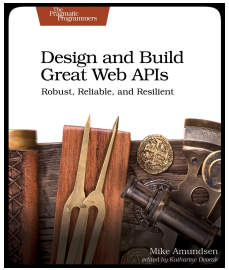


Exercise: Stand-Up

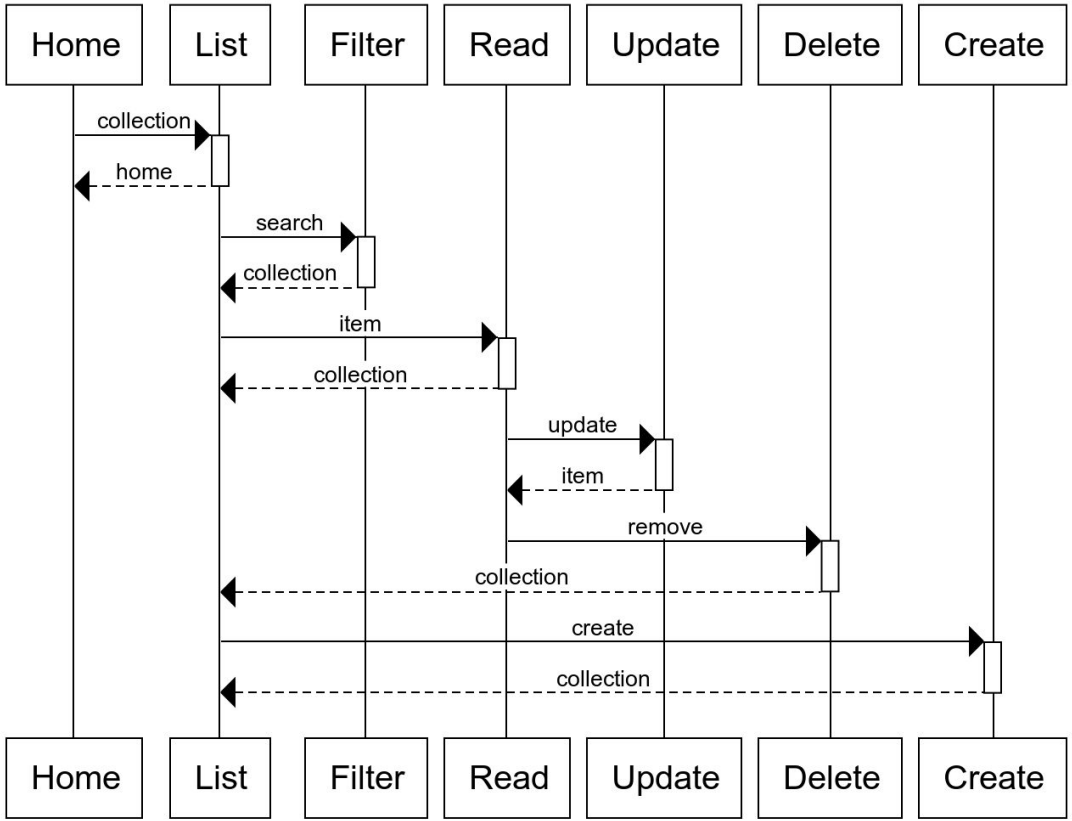


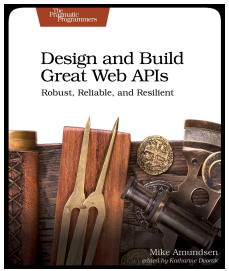
API Diagrams

- People "think" visually
- Diagrams reveal assumptions
- Diagrams are accessible
- Diagrams are easy to create/change/share



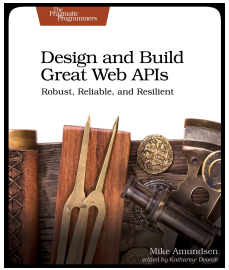
Company API Diagram





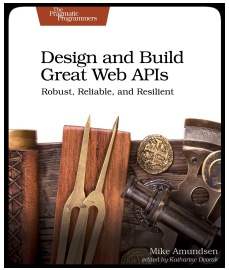
API Diagrams

- Web Sequence Diagram (WSD) documents
- WSD is a Fiat standard (shared but not documented)
- Many editors support it (VSCode, etc.)
- Several online host editors (plantuml.com, etc.)

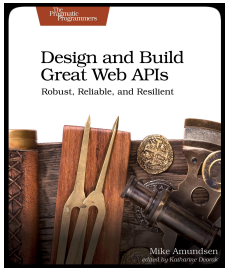


WSD file for Company Diagram

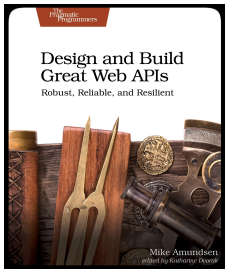
```
1  title Company API Diagram
2
3  Home->+List:collection
4  List-->-Home:home
5  List->+Filter:search
6  Filter-->-List:collection
7  List->+Read:item
8  Read-->-List:collection
9  Read->+Update:update
10 Update-->-Read:item
11 Read->+Delete:remove
12 Delete-->-List:collection
13 List->+Create:create
14 Create-->-List:collection
```



Diagrams make our solution visible

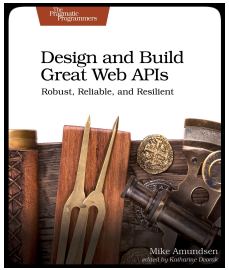


Exercise: Diagramming an API

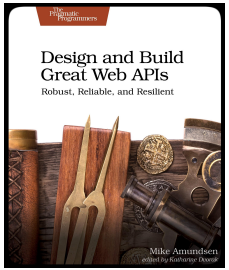


Diagramming an API

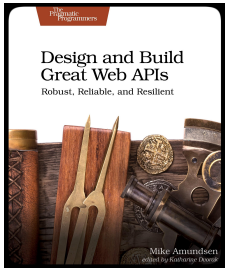
- Use the ToDo API story as a starter
- Produce an API Model document
  - Properties and Actions
- Create a WSD diagram
  - http://sequencediagrams.com



Exercise: Stand-Up

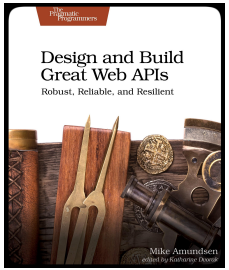


BREAK

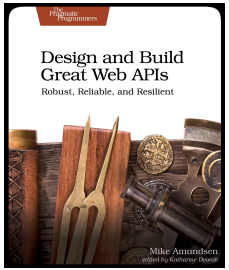


API Stories, Models and Diagrams

- The Importance of Stories
  - Every API starts with a story
- The Power of Models
  - Models help us focus on a solution
- The Value of Diagrams
  - Diagrams make our solution visible

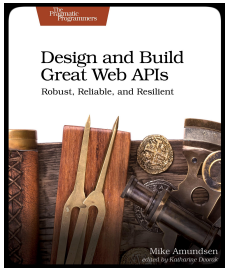


Your Assignment



Overnight Assignment for API Design

- Pick an API story as a starter
- Produce the default API Vocabulary Model
- Normalize the Model against schema.org
- Create a Sequence Diagram of the API Model



Designing Great APIs
Part One

@mamund
Mike Amundsen