

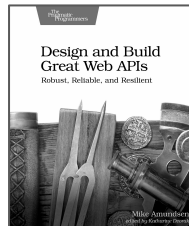
# Design and Build Great Web APIs

## Designing

@mamund

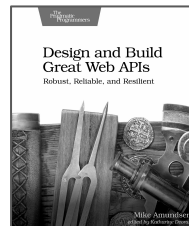
Mike Amundsen

[training.amundsen.com](http://training.amundsen.com)

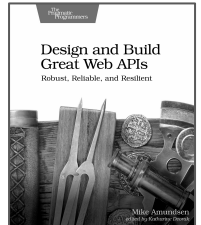


# Designing

- Document the Story
  - What is our purpose?
- Diagram the Flow
  - Make it visual
- Describe the API
  - This is not implementation

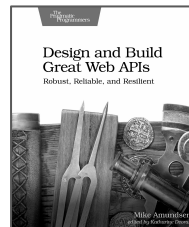


# Document the Story



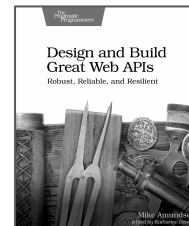
# Document the Story

- Every API has a purpose; a story
- Sometimes that story is hard to uncover
  - "We've always done it this way."
- Sometimes that story varies
  - "Well, that's not what WE need it for."
- Before designing and building, get the story straight



# Document the Story

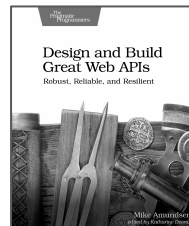
- Purpose
- Data
- Actions
- Processing



# Document the Story

## Purpose

We run a simple credit check periodically on companies that are our customers to update their baseline spending limit and discount percentages.

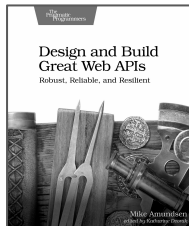


# Document the Story

## Purpose

### Data

Each company record has one or more `account` records. Each `account` record has a `spendingLimit` property and a `discountPercentage` property. By default these two values are set to `5000` and `5%`, respectively. However, we can modify those values by running a simple credit check using the customer name to see if we need to adjust those values up or down.



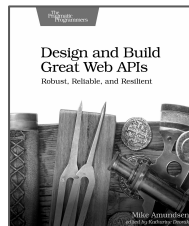
# Document the Story

## Purpose

## Data

## Actions

The credit check service has a safe method ( `checkCredit` ) which accepts a `companyName` and returns a `rating` (a single value between 1 and 10). That `rating` value can then be used to determine BigCo's standard `spendingLimit` and `discountPercentage` for that company's account record.





# Document the Story

## Purpose

## Data

## Actions

## Processing

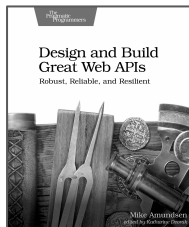
Each time the credit-check service is queried, it also writes a history record that can be called up later for review. History is recalled using the same `companyName` that was used when making the initial credit check. History records contain the following properties: `identifier`, `companyName`, `dateRequested`, and `rating`.

We run this credit-check when we first add a company account record and once a year after that (always on January 1st).



# Document the Story

- Interview each stakeholder
  - Record their references to data, actions, processing, purpose
- Use plain language
  - Speak in the language of th stakeholder, not the developer
- Document each story *\*separately\**
  - Share w/ each stakeholder for validation
- Write composite story to cover all stakeholders
  - Share w/ the group and work out details



# Document the Story

- Publish the agreed story
- Check it into the project repository
- Use it as a guide going forward

19 lines (11 sloc) | 1.48 KB

Raw Blame History

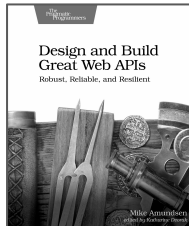
## Credit Check Story at BigCo, Inc.

### Purpose

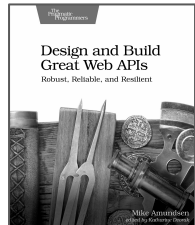
We run a simple credit check periodically on companies that are our customers to update their bsaseline spending limit and discount percentages.

### Data

Each company record has one or more `account` records. Each `account` record as a `spendingLimit` property and a `discountPercentage` property. By default these two values are set to `5000` and `5%`, respectively. However, we can modify those values by running a simple credit check using the customer name to see if we need to adjust those values up or down.

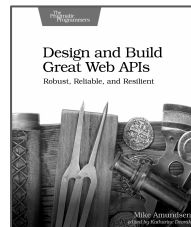


# Diagram the Flow



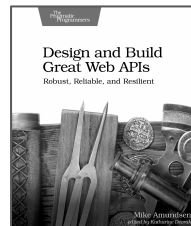
# Diagram the Flow

- Always include a visual representation of the API
- Diagram the action flow, not the resources or data
- Sequence diagrams are very easy

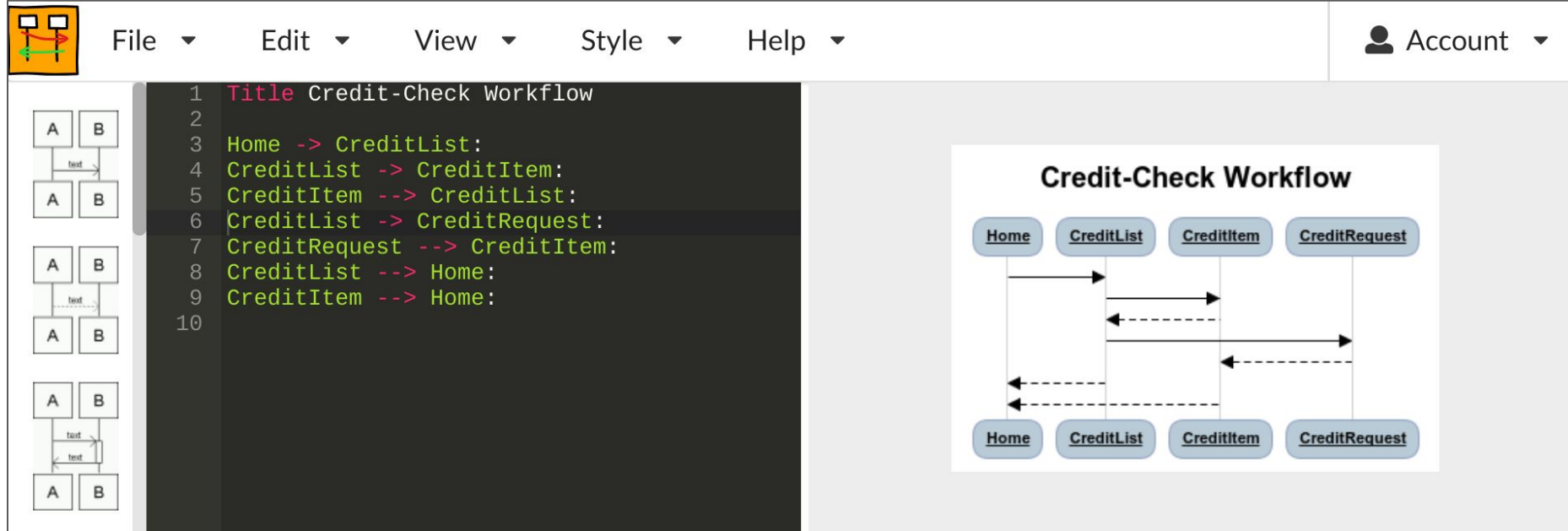


# Diagram the Flow

- The Web Sequence Diagram
- Write using the WSD text format
- Export PNG diagrams
- There is a handy API, too.



# Diagram the Flow : websequencediagrams.com



The screenshot shows the websequencediagrams.com interface. On the left is a toolbar with icons for creating lifelines, messages, and notes. The main editor area has a dark background with a sequence diagram titled "Credit-Check Workflow". The diagram consists of four lifelines: Home, CreditList, CreditItem, and CreditRequest. The messages are as follows:

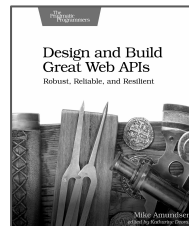
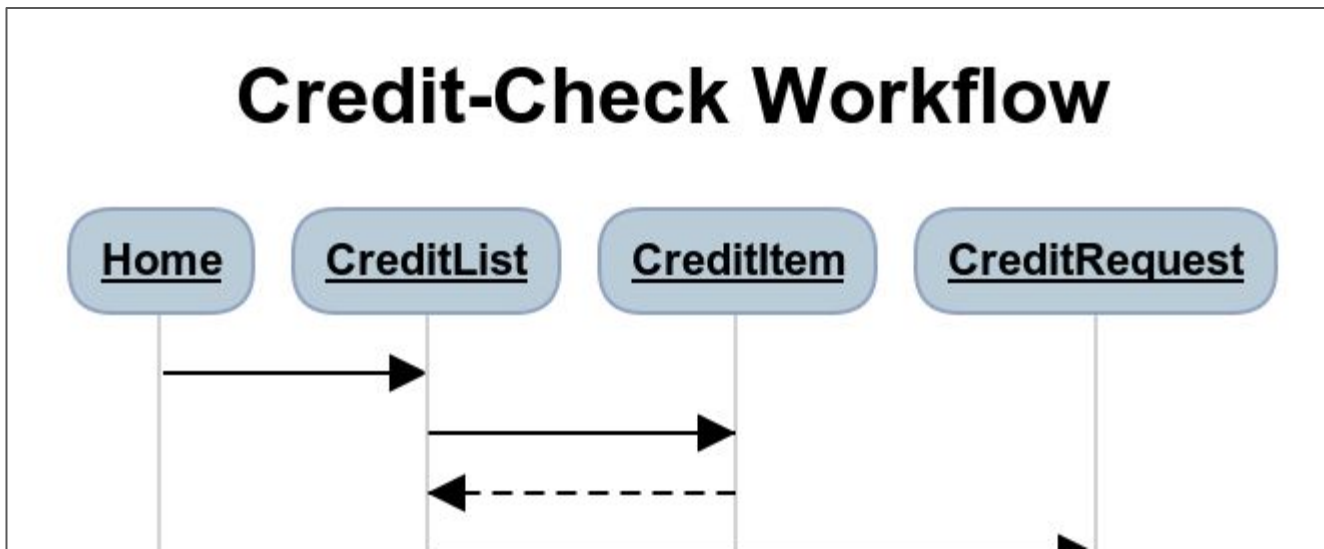
- Home -> CreditList: (solid arrow)
- CreditList -> CreditItem: (solid arrow)
- CreditItem --> CreditList: (dashed return arrow)
- CreditList -> CreditRequest: (solid arrow)
- CreditRequest --> CreditItem: (dashed return arrow)
- CreditItem --> CreditList: (dashed return arrow)
- CreditList --> Home: (dashed return arrow)

On the right side of the interface, there is a preview of the diagram on a white background. The title "Credit-Check Workflow" is centered above the lifelines. The lifelines are labeled "Home", "CreditList", "CreditItem", and "CreditRequest". The messages are represented by solid arrows for outgoing calls and dashed arrows for returns.



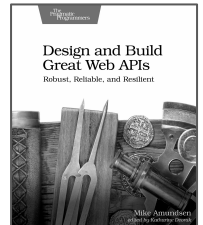
# Diagram the Flow

- Publish the final diagram
- Check it into the project repository
- Use it as a guide going forward



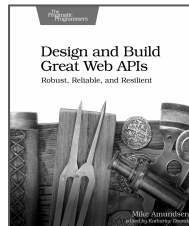


# Describe the API



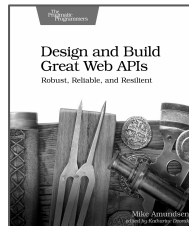
# Describe the API

- Design phase is not implementation phase
- API definitions are varied
  - WSDL, WADL, OpenAPI, AsyncAPI, protobuf, SDL
- Use an implementation-agnostic detailed description



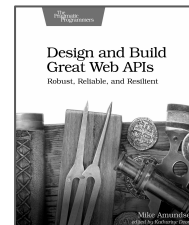
# Describe the API : ALPS

- Application-Level Profile Semantics
  - Amundsen-Richardson-Foster (2011)
- Identifies all interface properties
  - Id, familyName, givenName, telephone, etc.
- Identifies all interface actions
  - saveCompany, setStatus, approvePayroll, etc.
- Does not include implementation details
  - URLs, schemas, methods, response codes, etc.



# Describe the API : ALPS

```
alps:  
  version: '1.0'  
  description: ALPS document for BigCo Credit Check API
```



# Describe the API : ALPS

```
alps:
```

```
  version: '1.0'
```

```
  descri
```

```
    - id: creditCheckItem
```

```
      type: safe
```

```
      returns: '#ratingItem'
```

```
      descriptors:
```

```
        - href: '#id'
```

```
    - id: creditCheckForm
```

```
      type: unsafe
```

```
      returns: '#ratingItem'
```

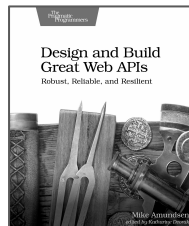
```
      descriptors:
```

```
        - href: '#id'
```

```
        - href: '#companyName'
```

```
        - href: '#ratingValue'
```

```
gCo Credit Check API
```



# Describe the API : ALPS

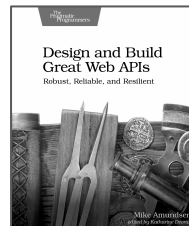
```
alps:
```

```
  version: '1.0'
```

```
  descri
```

- id: creditCheckItem  
 type: safe  
 returns: '#ratingItem'  
 descriptors:
  - href: '#id'
- id: creditCheckForm  
 type: unsafe  
 returns: '#ratingItem'  
 descriptors:
  - href: '#id'
  - href: '#companyName'
  - href: '#ratingValue'

- id: ratingItem  
 type: semantic  
 descriptors:
  - id: id  
 type: semantic
  - id: companyName  
 type: semantic
  - id: ratingValue  
 type: semantic
  - id: dateCreated  
 type: semantic
  - id: dateUpdated  
 type: semantic



# Describe the API : ALPS

- ALPS identifies
  - The state to pass in each message
  - The safety & idempotence of each action
- ALPS is part of Pivotal's Spring frameworks

## *Preface*

### Project Metadata

#### 1. Dependencies

## *Reference Documentation*

#### 2. Introduction

#### 3. Getting started

#### 4. Repository resources

#### 5. Paging and Sorting

#### 6. Domain Object Representations (Object Mapping)

## 12.1. Application-Level Profile Semantics (ALPS)

“ALPS is a data format for defining simple descriptions of application-level microformats. An ALPS document can be used as a profile to explain the application-agnostic media type (such as HTML, HAL, Collection+JSON, Sir) documents across media types.

— M. Admundsen / L. Richardson / M. Foster

<https://tools.ietf.org/html/draft-amundsen-richardson-foster-alps-00>

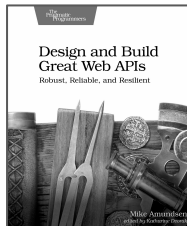
Spring Data REST provides an ALPS document for every exported repository. It contains transitions and the attributes of each repository.



# Describe the API : ALPS

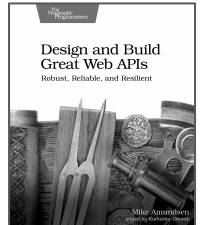
- Publish the profile
- Check it into the project repository
- Use it as a guide going forward

```
1  alps:
2    version: '1.0'
3    description: ALPS document for BigCo Onboarding API
4
5    descriptors:
6      - id: home
7        type: safe
8        returns: '#onboarding'
9
```





# Design Exercise

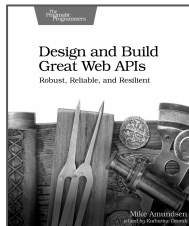


# Design Exercise

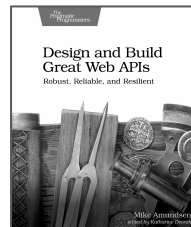
- Produce an ALPS document
  - Use one of the supplied stories (/stories/ folder)
- Purpose into ALPS description header
  - "We track customers at BigCo, Inc"
- Data into ALPS properties
  - `- id: customerId`
  - `- type: semantic`

## Actions into ALPS actions

- `- id: createCustomer`
- `- type: unsafe`

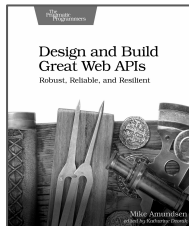


# Summary



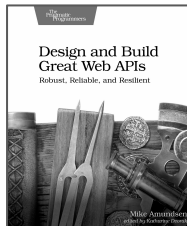
# Designing

- Document the Story
  - Purpose, Actions, & Data
- Diagram the Flow
  - Sequence diagrams are not HTTP
- Describe the API
  - Before OpenAPI, AsyncAPI, protobuf and SDL

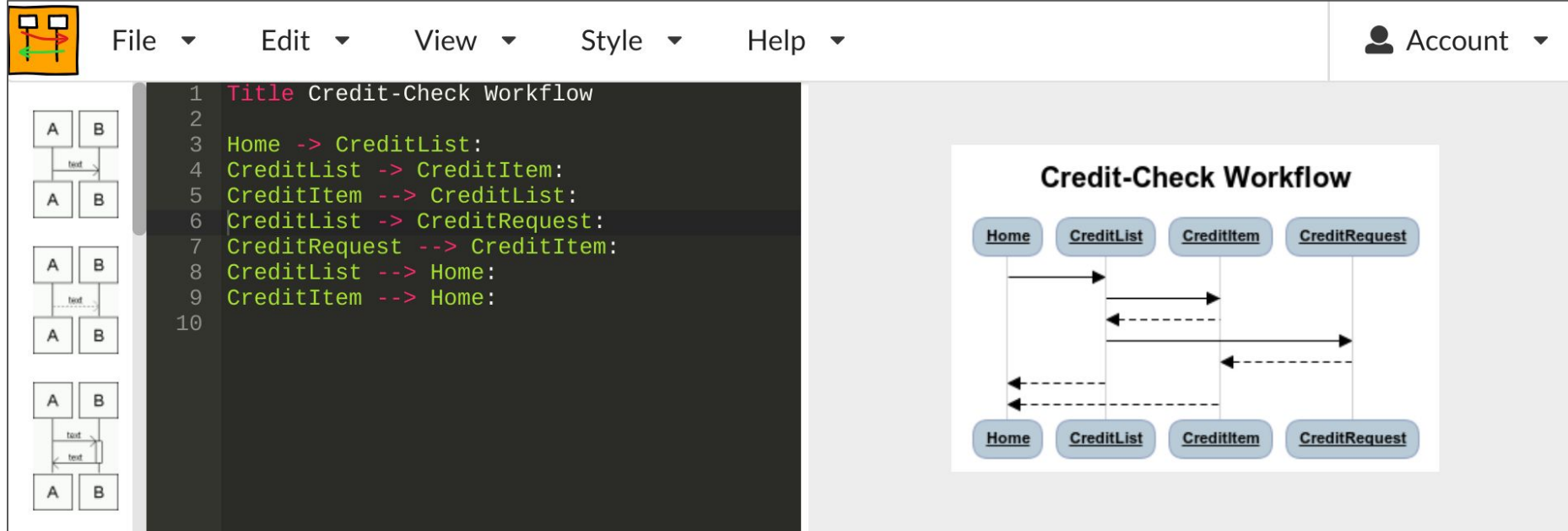


# Document the Story

- Interview each stakeholder
  - Record their references to data, actions, processing, purpose
- Use plain language
  - Speak in the language of th stakeholder, not the developer
- Document each story *\*separately\**
  - Share w/ each stakeholder for validation
- Write composite story to cover all stakeholders
  - Share w/ the group and work out details



# Diagram the Flow : websequencediagrams.com



The screenshot shows the websequencediagrams.com interface. The top navigation bar includes a logo, a menu (File, Edit, View, Style, Help), and an Account link. The main workspace is divided into three sections: a palette on the left with lifelines A and B, a code editor in the center, and a preview area on the right.

The code editor contains the following sequence diagram definition:

```
1 Title Credit-Check Workflow
2
3 Home -> CreditList:
4 CreditList -> CreditItem:
5 CreditItem --> CreditList:
6 CreditList -> CreditRequest:
7 CreditRequest --> CreditItem:
8 CreditList --> Home:
9 CreditItem --> Home:
10
```

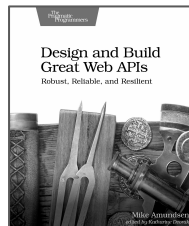
The preview area displays the rendered sequence diagram titled "Credit-Check Workflow". It shows four lifelines: Home, CreditList, CreditItem, and CreditRequest. The interactions are as follows:

- Home sends a message to CreditList.
- CreditList sends a message to CreditItem.
- CreditItem returns a message to CreditList.
- CreditList sends a message to CreditRequest.
- CreditRequest returns a message to CreditItem.
- CreditList returns a message to Home.
- CreditItem returns a message to Home.



# Describe the API : ALPS

- Application-Level Profile Semantics
  - Amundsen-Richardson-Foster (2011)
- Identifies all interface properties
  - Id, familyName, givenName, telephone, etc.
- Identifies all interface actions
  - saveCompany, setStatus, approvePayroll, etc.
- Does not include implementation details
  - URLs, schemas, methods, response codes, etc.



# Design and Build Great Web APIs

## Designing

@mamund

Mike Amundsen

[training.amundsen.com](http://training.amundsen.com)

