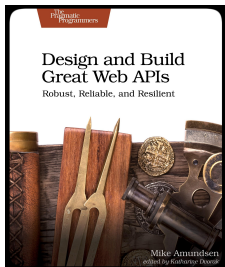


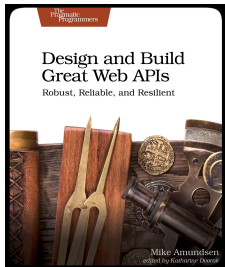
Building Great Web APIs

Part Two

@mamund
Mike Amundsen



Welcome Back

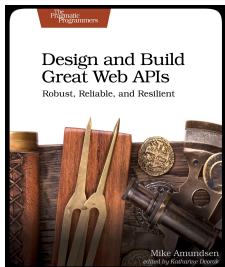


copyright © 2020 - amundsen.com, Inc.

Three-Phase API Implementation



- To reduce cost and risk, take a three-phase approach
- **Sketching** - disposable experiments
- **Prototyping** - testable examples
- **Building** - production implementation



Prototyping with OpenAPI

The screenshot displays the SwaggerHub interface for prototyping an OpenAPI. The top navigation bar includes the SwaggerHub logo, user profile 'mamund', and communication icons. The main header shows the project name 'Hand-todo' and version '1.0.0'. On the right, there are icons for document, settings, notifications, and sharing, along with an 'Export' dropdown.

The left sidebar contains icons for a user profile, code editor, and document. The main editor area shows the OpenAPI specification in a dark-themed code editor. The specification includes the OpenAPI version (3.0.1), a title, a description, a server URL, and a single endpoint for GET /todo.

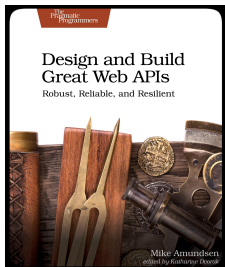
```
1 openapi: 3.0.1
2
3 # Added by API Auto Mocking Plugin
4 servers:
5   - description: SwaggerHub API Auto Mocking
6     url: https://virtserver.swaggerhub.com/amundsen/Hand-todo/1.0.0
7 info:
8   title: simpleTodo
9   description: Simple Todo list example
10  version: 1.0.0
11
12 paths:
13   /todo:
14     get:
15       summary: 'return list of todo items'
16       operationId: todoList
17       responses:
18         200:
19           description: get todoList
20           content:
21             application/json:
22               schema:
23                 type: array
24                 items:
25                   type: object
```

The right sidebar shows the 'default' section with four API endpoints, each with a color-coded button for the HTTP method:

- GET /todo** return list of todo items
- POST /todo** create a new todo item
- GET /todo/{id}** return todo item
- DELETE /todo/{id}** remove a single todo item

Below the endpoints, the 'Schemas' section is visible, showing a schema for 'todoItem'.

Building with NodeJS/Express & DARRT



```
mca@penguin:~/projects/api-tool-kit/api-starter-kit/darrrt$ tree
```

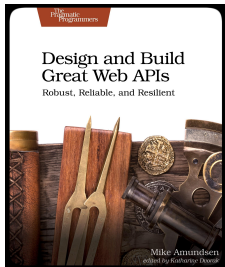
```
├── actions.js
├── data.js
├── lib
│   ├── component.js
│   ├── ejs-helpers.js
│   ├── storage.js
│   └── utils.js
├── representation.js
├── representors
│   ├── app-json.js
│   ├── forms-json.js
│   ├── links-json.js
│   ├── prag-json.js
│   └── text-csv.js
├── resources.js
└── transitions.js
```

```
2 directories, 14 files
```

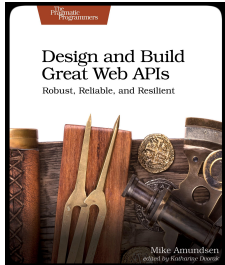
```
mca@penguin:~/projects/api-tool-kit/api-starter-kit/darrrt$
```

Overnight Assignment for API Design

- Start from an ALPS Description
- Using the API Starter Kit...
 - Update `data.js`, `actions.js`, & `resources.js` as needed
- Use `npm run dev` to validate the API Project



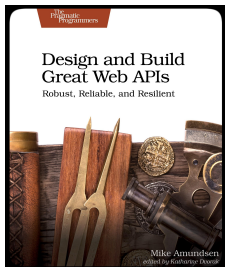
Assignment: Review



copyright © 2020 - amundsen.com, Inc.

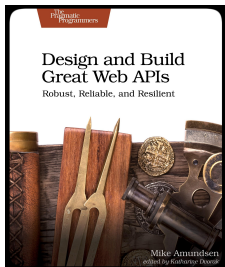
ALPS API Description

```
1 #####
2 # TODO .....: Application-Level Profile Semantics doc
3 # Author ...: Mike Amundsen (@mamund)
4 # Date .....: 2020-04-13
5 #####
6
7 alps:
8   version: '1.0'
9   doc:
10     value: ALPS document for BigCo Company API
11
12   # metadata
13   name: Company API
14   id: http://alps.io/profiles/mamund/company
15   root: http://api.example.org/company
16
17   descriptor:
18     # properties
19     - id: id
20       type: semantic
21
22     - id: status
23       type: semantic
24       text: 'suspended, pending, active, closed'
25
```



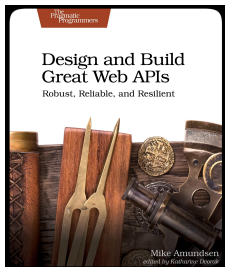
DARRT : data.js

```
1 //
2 // this service's message properties
3 exports.props = [
4   'id',
5   'companyName',
6   'streetAddress',
7   'city',
8   'stateProvince',
9   'postalCode',
10  'country',
11  'telephone',
12  'email',
13  'status',
14  'dateCreated',
15  'dateUpdated'
16 ];
17
18 // required properties
19 exports.reqd = ['id', 'companyName', 'email', 'status'];
20
21 // enumerated properties
22 exports.enums = [
23   {status: ['pending', 'active', 'suspended', 'closed']}
24 ];
```



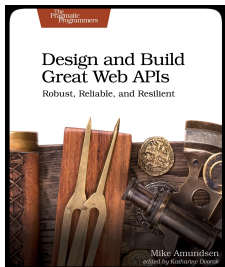
DARRT : actions.js

```
1 /*****
2 // bigco, inc
3 // company action elements
4 // 2020-02-01 : mamund
5 *****/
6
7 var component = require('./lib/component');
8 var data = require('./data');
9
10 /*****
11 // actions for the company service
12 // home, create, list, filter,
13 // read, update, status, remove
14 *****/
15
16 module.exports.home = function(req,res) {}
17 module.exports.create = function(req,res) {}
18 module.exports.list = function(req,res) {}
19 module.exports.filter = function(req,res) {}
20 module.exports.read = function(req,res) {}
21 module.exports.update = function(req,res) {}
22 module.exports.status = function(req,res) {}
23 module.exports.remove = function(req,res) {}
24
```

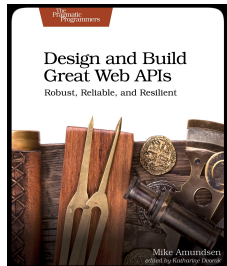


DARRT : resources.js

```
1 /*****
2 // bigco, inc
3 // company resources
4 *****/
5
6 var express, router, bodyParser, actions, representation,
7     transitions, utils, templates, forms, metadata;
8
9 init();
10
11 // shared metadata for this service
12 metadata = [
13   {name: "title", value: "BigCo Company Records"},
14   {name: "author", value: "Mike Amundsen"},
15   {name: "release", value: "1.0.0"}
16 ];
17
18 router.get('/', function(req, res){...});
19 router.post('/', function(req, res){...});
20 router.get('/list/', function(req, res){...});
21 router.get('/filter/', function(req, res){...});
22 router.get('/:id', function(req, res){...});
23 router.put('/:id', function(req, res){...});
24 router.delete('/:id', function(req, res){...});
25 router.patch('/status/:id', function(req, res){...});
```

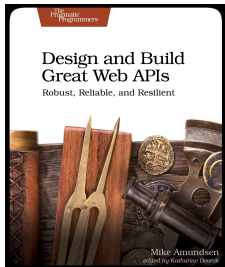


BREAK



copyright © 2020 - amundsen.com, Inc.

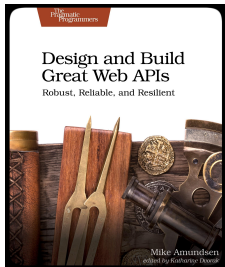
Validating the API



copyright © 2020 - amundsen.com, Inc.

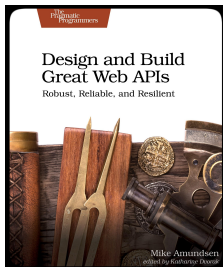
Validating the API

- Compose simple "curl tests" to validate the API
- Test each endpoint
- Confirm "happy path" tests (200 OK)
- Include "sad path" tests (400 Bad Request)

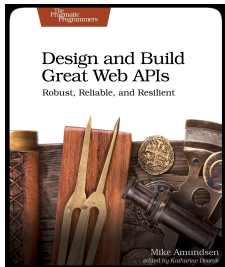


Validating the API

```
1 # company SRIS
2 # 2020-03 mamund
3
4 # happy
5 # these should return 200
6 http://localhost:8484/
7 http://localhost:8484/list/
8 http://localhost:8484/filter?status=active
9 http://localhost:8484/ -X POST -d id=qlw2e3r4&status=pending&companyName=MikeCo&en
10 http://localhost:8484/qlw2e3r4 -X PUT -d streetAddress=123%
    20Main&city=Byteville&stateProvince=MD&postalCode=12345&country=USA&telephone=2345
11 http://localhost:8484/status/qlw2e3r4 -X PATCH -d status=active
12 http://localhost:8484/qlw2e3r4 -X DELETE
13
14 # sad
15 # these should return 400
16 http://localhost:8484/xxxx -X GET
17 http://localhost:8484/12345 -X DELETE
18 http://localhost:8484/ -X POST -d id=12345
19 http://localhost:8484/ -X POST -d id=12345&companyName=BadRec,%20Inc.
20 http://localhost:8484/ -X POST -d id=12345&companyName=BadRec,%20Inc.&email=badrec
21 http://localhost:8484/ -X POST -d id=12345&companyName=BadRec,%
    20Inc.&email=badrec@example.org&status=broken
22 http://localhost:8484/ -X POST -d id=12345&companyName=BadRec,%
    20Inc.&email=badrec@example.org&status=pending
23 http://localhost:8484/ -X POST -d id=12345&companyName=BadRec,%
    20Inc.&email=badrec@example.org&status=pending
```

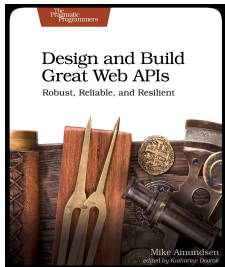


Exercise: Write Validation Calls

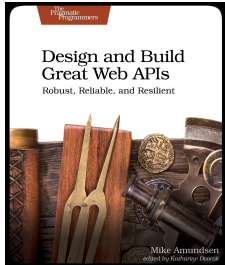


Writing Validating Calls

- Use curl
- Write out a 'happy path' call for each endpoint
- Write out a 'sad path' call to confirm data.js rules

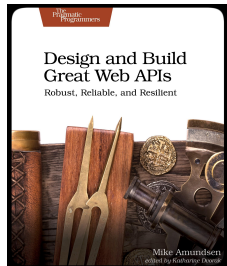


Exercise: Stand-Up



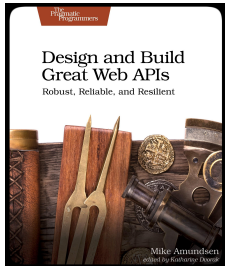
copyright © 2020 - amundsen.com, Inc.

BREAK



copyright © 2020 - amundsen.com, Inc.

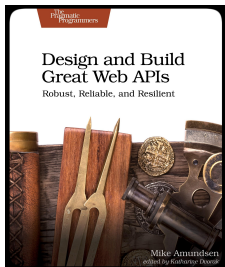
Deploying APIs



copyright © 2020 - amundsen.com, Inc.

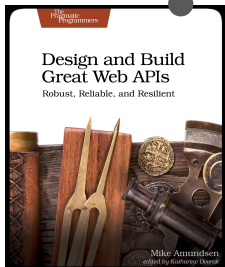
Deploying APIs

- Challenges of Deployment
- Git-based Deployment
- Using **Heroku**



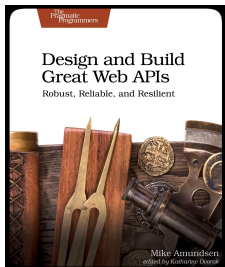
Deploying APIs - Challenges

- Deploying your app can be complicated
- Compatibility
 - Hardware
 - OS
 - Platform
 - Framework
- Dependencies



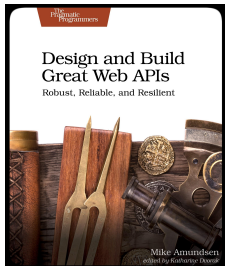
Deploying APIs - DevOps

- DevOps was created to help with all this
- Developers & Operators working together
- Started as a hashtag on twitter #DevOps
- Series of small conferences started in 2009
- Emphasis on automation to improve reliability



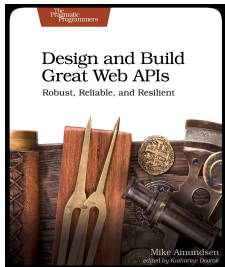
Deploying APIs - Tools

- Build tools
- CI/CD pipeline
- Docker (containers)
- Kubernetes (deployment orchestration)



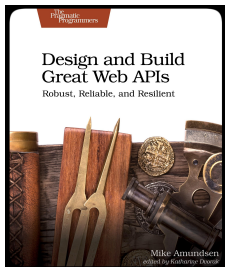
Deploying APIs - Using Heroku

- Cloud platform (2007)
- Originally just for Ruby/Rails projects
- Now supports Java, NodeJS, Python, Go, Clojure, Scala
- Acquired by Salesforce in 2011
- Full platform w/ marketplace ecosystem
- Heroku uses proprietary container tech (Dynos)



Deploying APIs - Using Heroku

- Download CLI
<https://devcenter.heroku.com/articles/heroku-cli>
- Documentation
<https://devcenter.heroku.com/articles/using-the-cli>
- Create an Account (required)
<https://signup.heroku.com/>



Deploying APIs - Using Heroku

- Git deploy tutorial

<https://devcenter.heroku.com/articles/git>

The `heroku create` CLI command creates a new empty application on Heroku, along with an associated empty Git repository. If you run this command from your app's root directory, the empty Heroku Git repository is automatically set as a remote for your local repository.

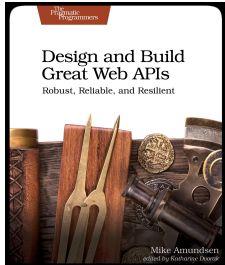
```
$ heroku create
Creating app... done, ⬢ thawing-inlet-61413
https://thawing-inlet-61413.herokuapp.com/ | https://git.heroku.com/thawing-inlet-61413.git
```

You can use the `git remote` command to confirm that a remote named `heroku` has been set for your app:

```
$ git remote -v
heroku https://git.heroku.com/thawing-inlet-61413.git (fetch)
heroku https://git.heroku.com/thawing-inlet-61413.git (push)
```

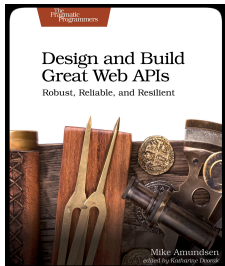


Releasing Exercise

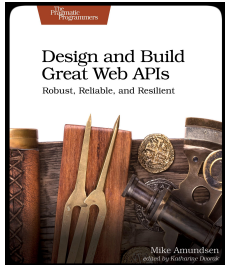


Releasing Exercise

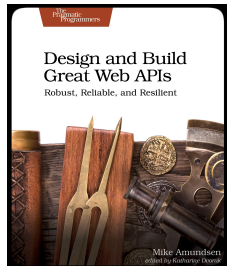
- Open command window in your project
- `heroku login`
- `heroku create`
- `git push heroku master`



Exercise: Review

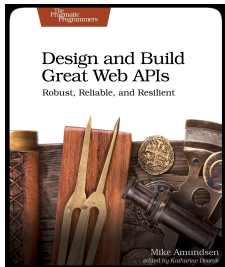


BREAK



copyright © 2020 - amundsen.com, Inc.

Building APIs: Summary

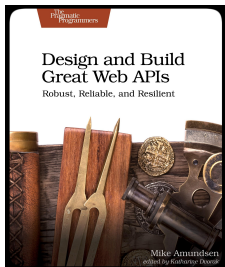


copyright © 2020 - amundsen.com, Inc.

Three-Phase API Implementation

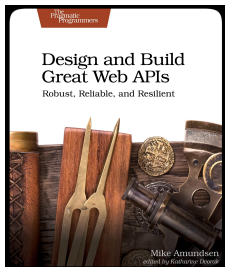


- To reduce cost and risk, take a three-phase approach
- **Sketching** - disposable experiments
- **Prototyping** - testable examples
- **Building** - production implementation



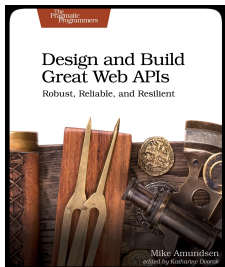
Building APIs : DARRT

- Simple process for publishing running interfaces
- Data
- Actions
- Resources
- Representations
- Transitions



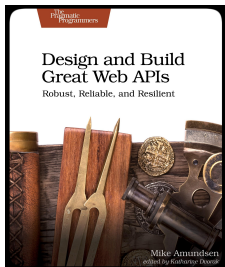
Building APIs : DARRT

- Simple process for publishing running interfaces
- Data
- Actions
- Resources
- Representations
- Transitions

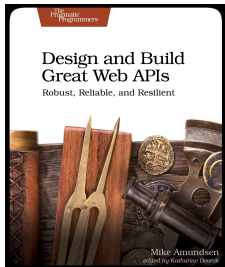


Deploying APIs

- Challenges of Deployment
- Git-based Deployment
- Using **Heroku**



Open Question Time



Building Great Web APIs

Part Two

@mamund
Mike Amundsen

