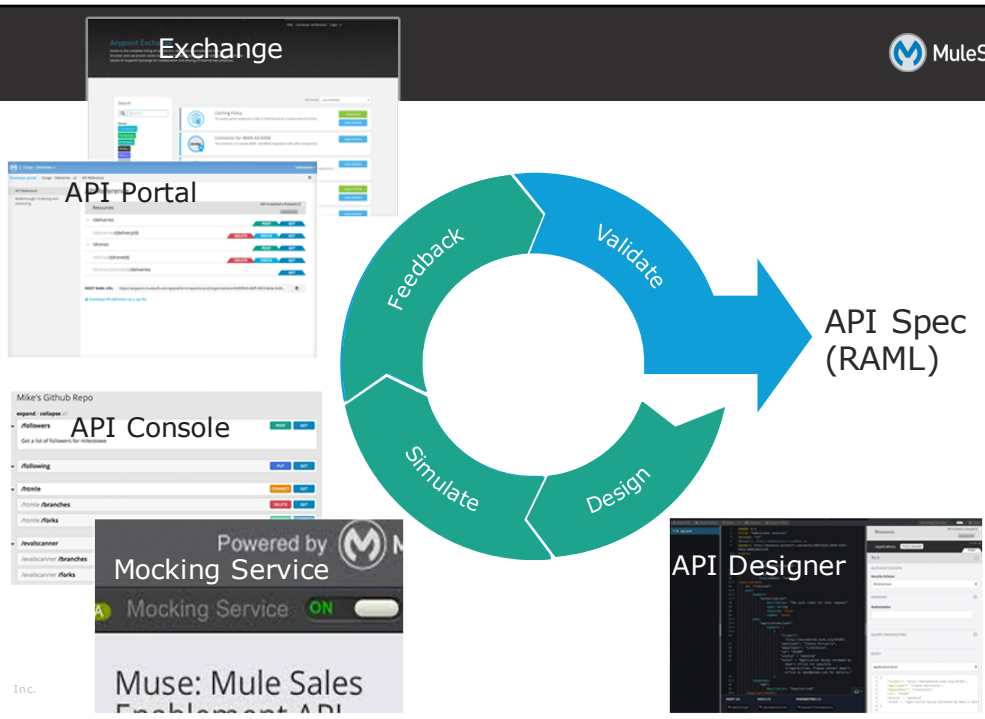
Module 2: Designing APIs

Spec driven development

- We discussed last module about the benefits of designing an API first before actually building it
- This is often referred to as spec driven development
  - A development process where your application is built in two distinct phases
    - The creation of a spec (the design phase)
    - Development of code to match the spec (the development phase)
- In this module, we'll
  - Create this spec (the API definition) using a standardized API description language (RAML)
  - Then learn to test it with users without writing any code

Goal



Objectives

- Define an API with RAML, the Restful API Modeling Language
- Mock an API to test its design before it is built
- Create a portal for developers to learn how to use an API
- Make an API discoverable by adding it to the private Exchange

Introducing options for defining APIs

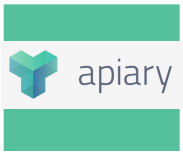
Approaches to API design

MuleSoft

HAND CODING

SWAGGER

BLUEPRINT



RAML

Introducing RAML

RAML: RESTful API Modeling Language

MuleSoft

- A simple, structured, and succinct way of describing RESTful APIs
  - The resources
  - The HTTP methods that can be used for each resource
  - Any method request parameters and their data type
  - The response types and sample responses
  - Schemas and more!

RAML

http://raml.org

- Developed to help out the current API ecosystem
  - Encourages reuse, enables discovery and pattern-sharing, and aims for merit-based emergence of best practices
- A non-proprietary, vendor-neutral open spec

RAML files can be used to …

MuleSoft

- Auto-generate API documentation
  - For an API Console in an API Portal (interactive docs)
  - Using hundreds of other tools: http://raml.org/developers/document-your-api
- Generate mocked endpoints so an API can be interactively tested before it is built
  - In an API Console
  - Using popular testing tools: http://raml.org/developers/test-your-api
- Auto-generate an implementation interface with sever-side generators In Mule, using APIkit
  - In NodeJS, Java, .NET, Python…: http://raml.org/developers/build-your-api
- To enable auto-discovery of endpoints for users in tools like Studio

RAML syntax

MuleSoft

- RAML is based on broadly-used standards such as YAML and JSON
- Uses a human-readable data serialization format where **data structure hierarchy is specified by indentation**
  - Not additional markup characters

```
#%RAML 0.8
title: American Flights API
baseUri: http://apdev-american-ws.cloudhub.io/api
version: 1.0

/flights:
  get:
  post:

  /{ID}:
    get:
    delete:
    put:
      responses:
        200:
          body:
            application/json:
```

Notice the indentation used to specify to what each line applies

Defining resources and methods

MuleSoft

- Resources are the objects identified by the web service URL that you want to act upon using the HTTP method used for the request
- All resources begin with a slash
- Any methods and parameters nested under a resource belong to and act upon that resource
- Nested resources are used for a subset of a resource to narrow it
  - URI parameter are enclosed in {}

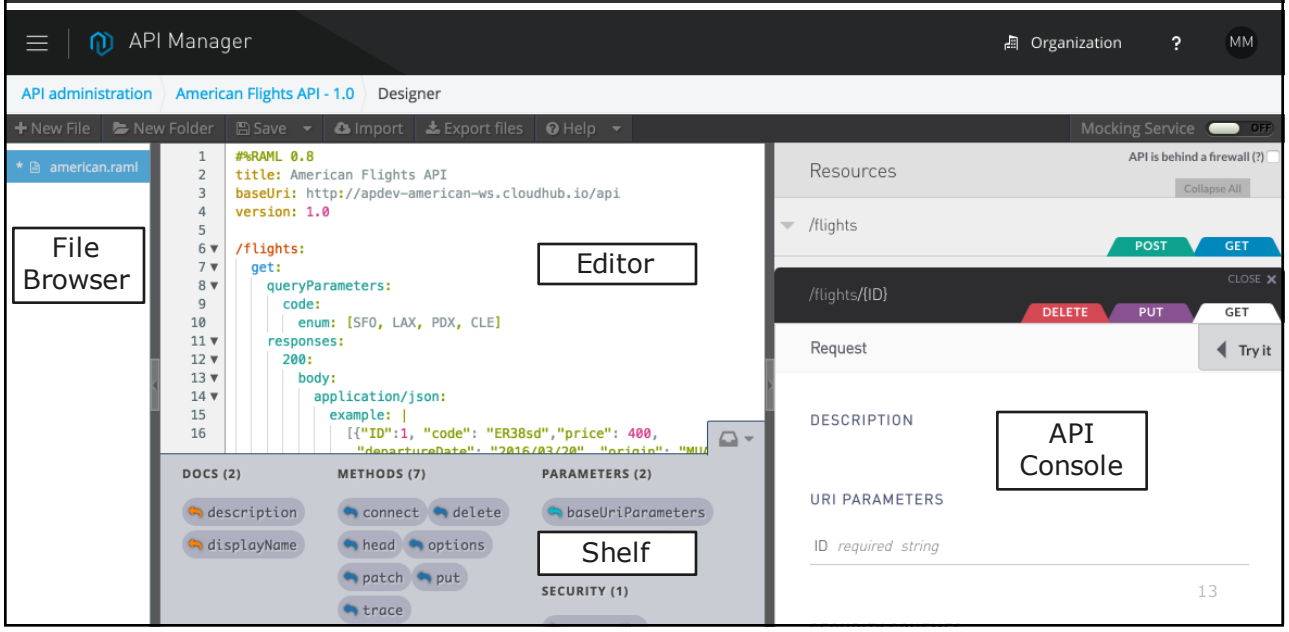
```
#%RAML 0.8
title: American Flights API
baseUri: http://apdev-american-ws.cloudhub.io/api
version: 1.0

/flights:
  get:
  post:

  /{ID}:
    get:
    delete:
    put:
      responses:
        200:
          body:
            application/json:
```

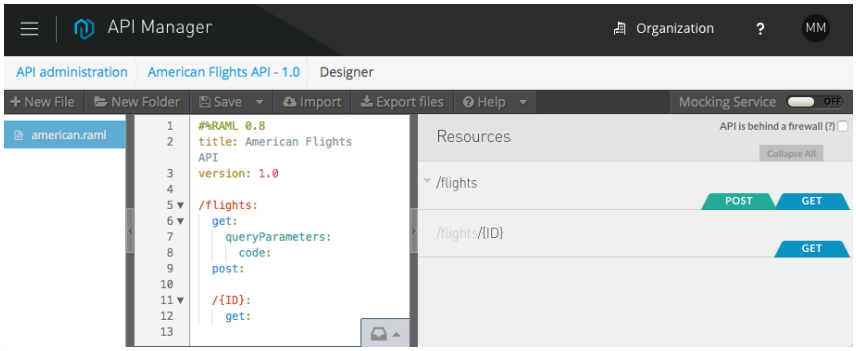

Using API Designer to define APIs with RAML

The API Designer



Walkthrough 2-1: Use API Designer to define an API with RAML

- Define resources and nested resources
- Interact with the API using the API Console
- Define get and post methods
- Specify query parameters

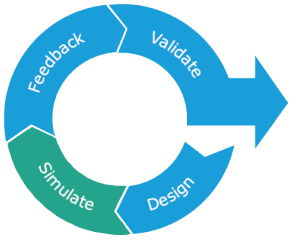


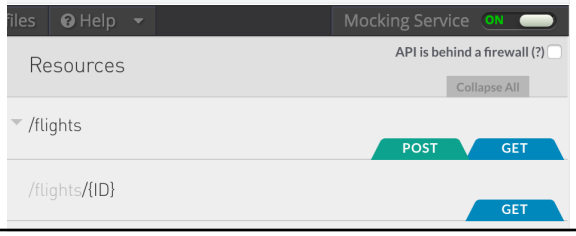
Testing API design without writing code

Simulating an API

MuleSoft

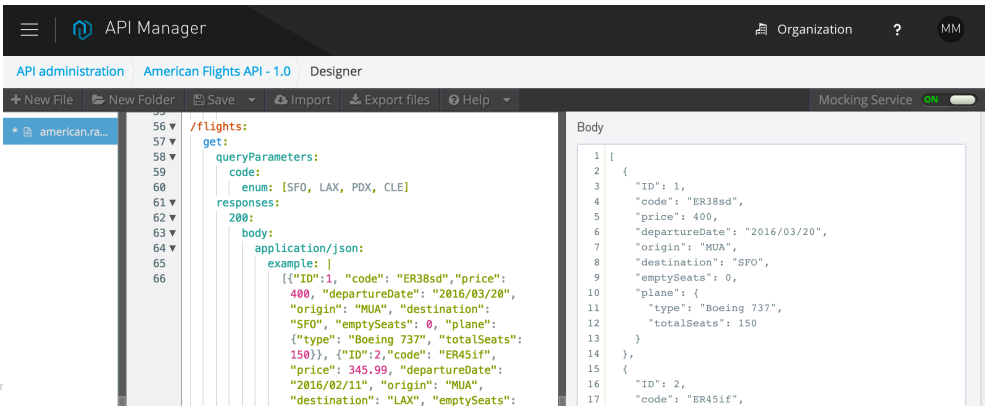
- You can mock an API to test it before it is implemented
  - Useful to get early feedback from developers
- Use the API Console and the mocking service to run a live simulation
- The API Console is available in
  - API Designer – so the API designer can test it
  - Anypoint API Portals – so users/developers can test it





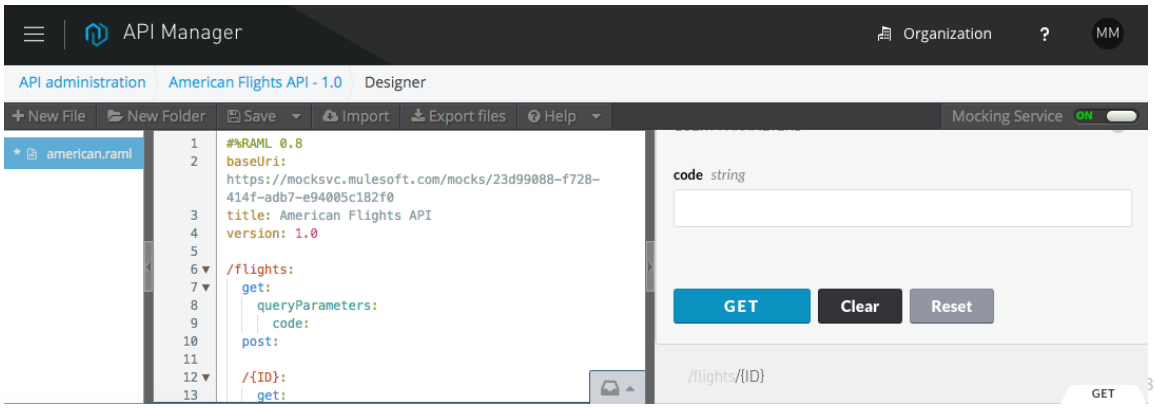
Using the mocking service

- The mocking service enables the API Console be used to make calls to an API that has no implementation yet
  - It returns sample API responses defined in the API definition
  - Can be used in the API Designer and in API portals



Walkthrough 2-2: Use the mocking service to test an API

- Add example responses to the API definition
- Turn on the mocking service
- Use the API Console to make calls to the mocked API



Using RAML to define specifications for requests and responses

Defining method response details with RAML

MuleSoft

- Responses must be a map of one or more HTTP status codes
- For each response, specify possible return data types along with descriptions, examples, or schemas

```
/flights:
  get:
    queryParameters:
      code:
        enum: [SFO, LAX, PDX, CLE]
    responses:
      200:
        body:
          application/json:
            example: |
              [{"ID":1, "code": "ER38sd","price": 400, "departureDate": "2016/03/20",
               "origin": "MUA", "destination": "SFO", "emptySeats": 0, "plane": {"type":
               "Boeing 737", "totalSeats": 150}}, {"ID":2,"code": "ER45if", "price":
               345.99, "departureDate": "2016/02/11", "origin": "MUA", "destination":
               "LAX", "emptySeats": 52, "plane": {"type": "Boeing 777", "totalSeats":
               300}}]
```

Notice the pipe symbol (|), which allows spanning a string scalar across multiple lines

All

20

Defining method request details with RAML

MuleSoft

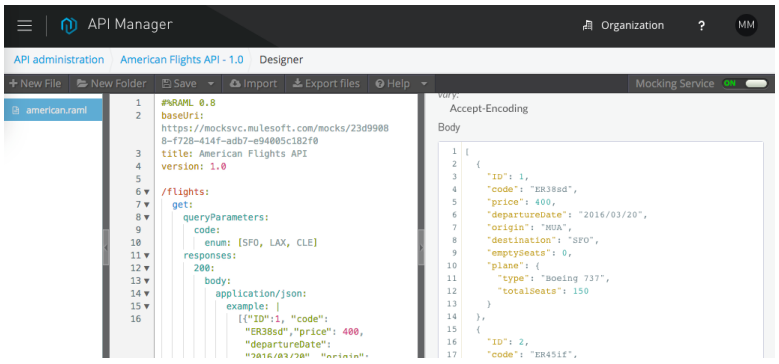
- For a request, similarly specify the possible request data types along with descriptions, examples, or schemas

```
/flights:
  get:↔
  post:
    body:
      application/json:
        example: |
          {"code": "ER38sd","price": 400, "departureDate":
           "2016/03/20", "origin": "MUA", "destination": "SFO",
           "emptySeats": 0, "plane": {"type": "Boeing 737",
           "totalSeats": 150}}
    responses:
      201:
        body:
          application/json:
            example: |
              {"message": "Flight added (but not really)"}
```


Walkthrough 2-3: Add request and response details

MuleSoft

- Specify data types for GET and POST method responses
- Add example JSON responses
- Specify possible query parameter values
- Test the API and get the sample responses

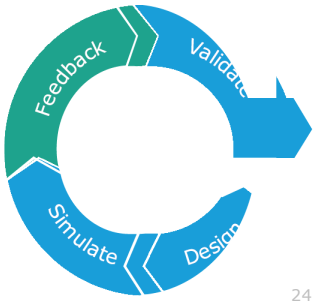


Engaging with users

Engaging users during the API design phase

MuleSoft

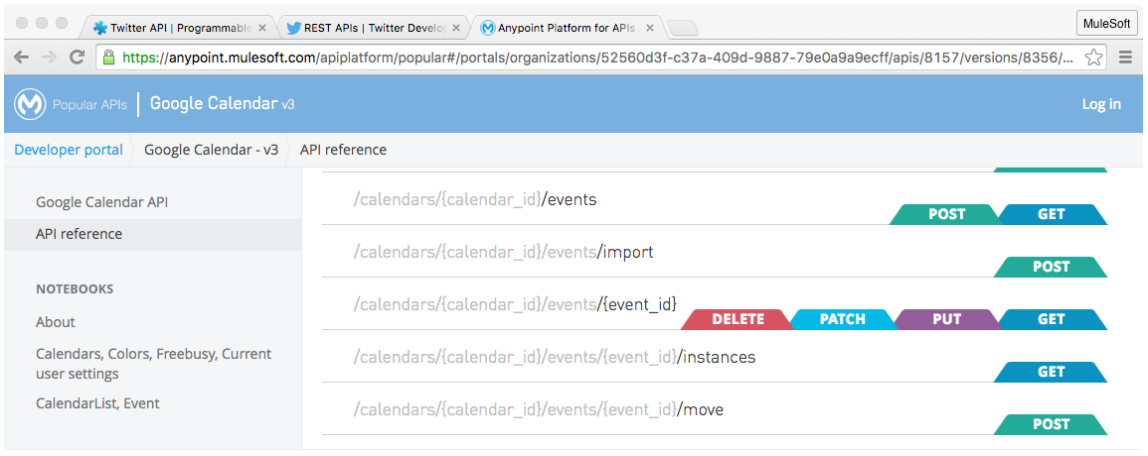
- To build a successful API, you should define it iteratively
  - Get feedback from developers on usability and functionality along the way
- To do this, you need to provide ways for developers to discover and play with the API
- Anypoint Platform makes this easy with API portals and the Exchange
  - Make an API portal to let users learn about an API and try it out
  - Add the API to the public Exchange or your private Exchange so users can discover it



Creating an API portal

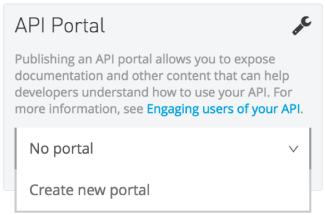
An Anypoint API Portal

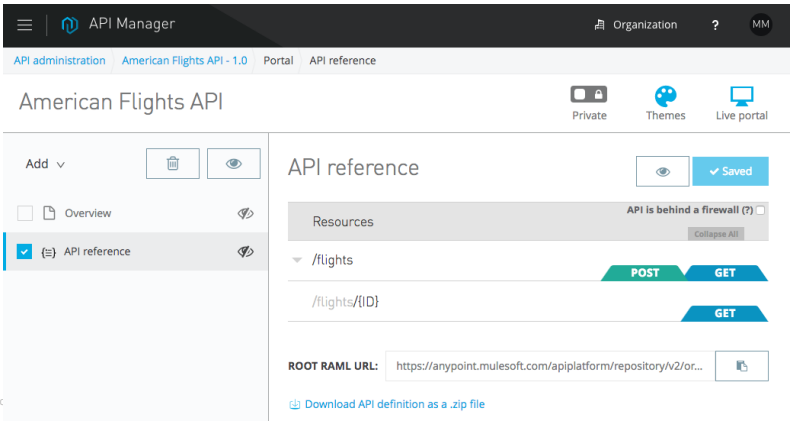
MuleSoft



Creating Anypoint API Portals

- API portals can be auto-generated for an API from the API Manager
- Launches the API Portal Designer





27

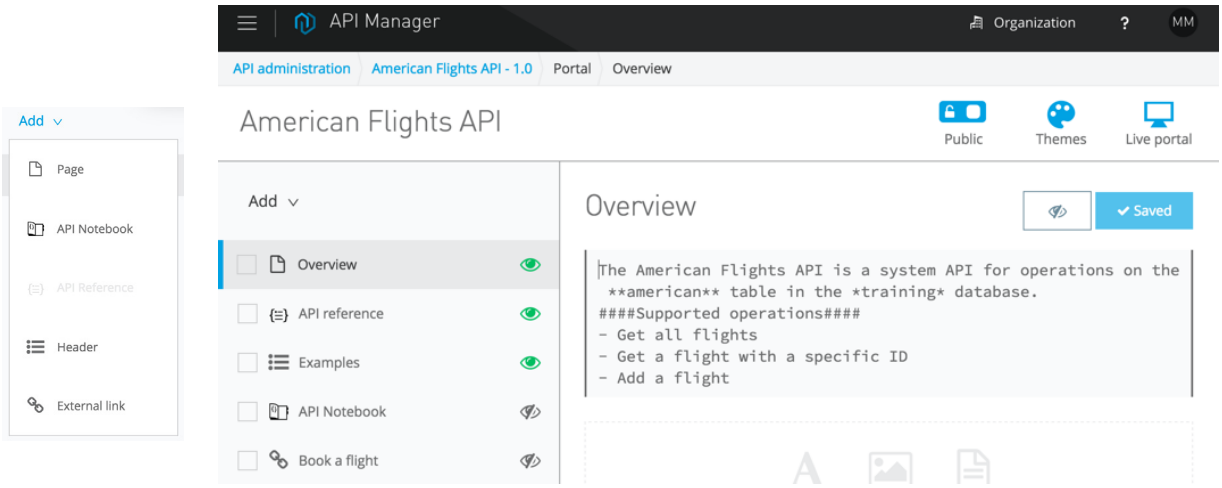
Customizing API portals

- Starting portal has an API reference page with an API Console
- You use the API Portal Designer to
  - Add text to the home page
    - Text is written in GitHub flavored markdown (https://help.github.com/categories/writing-on-github/)
  - Add additional content for tutorials, code snippets, and examples
    - To the left-side navigation, add pages, API Notebooks, headers, and external links
    - An API Notebook is a workspace for executing JavaScript commands for testing chained API calls for common use cases and workflows
  - Skin the portal
  - Make the portal public, private, or both

28

API Portal Designer

MuleSoft

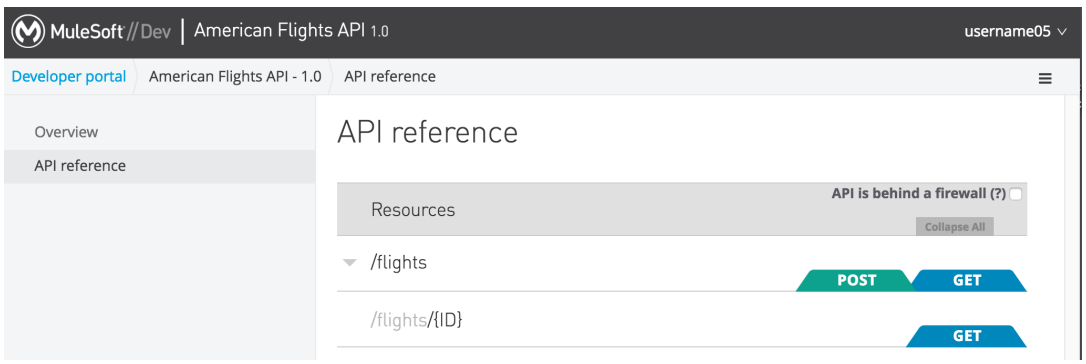


29

Walkthrough 2-4: Create an API portal

MuleSoft

- Create an API portal
- Add content to the portal
- Customize the portal
- View the resulting developer portal



30

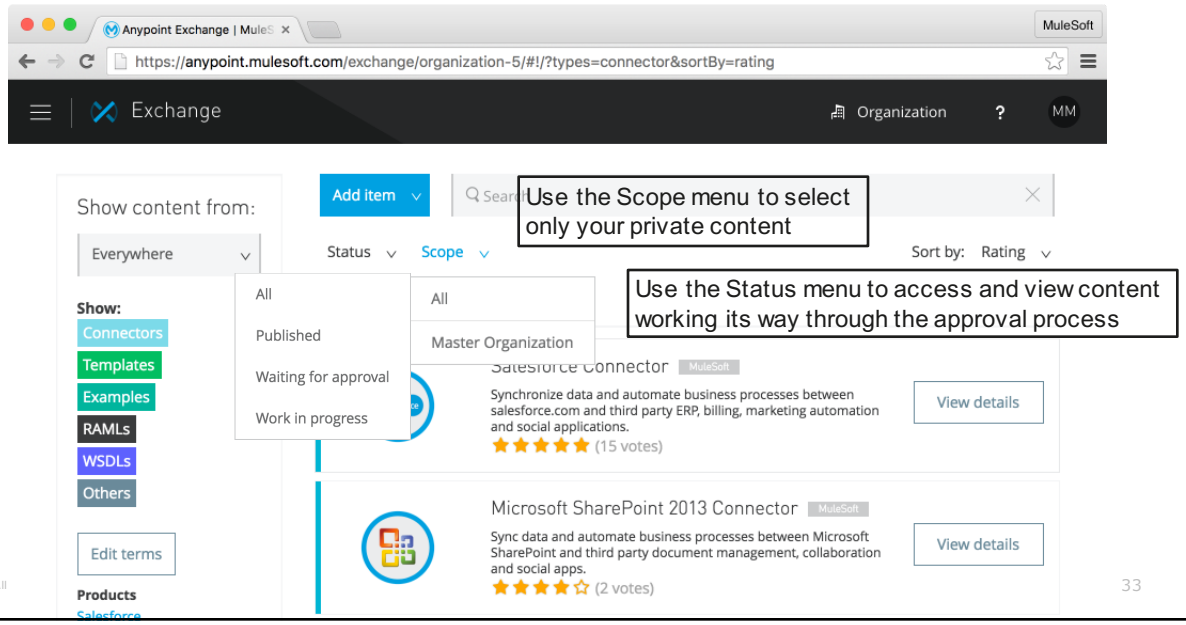
Adding an API to the Anypoint Exchange

Anypoint Exchange

MuleSoft

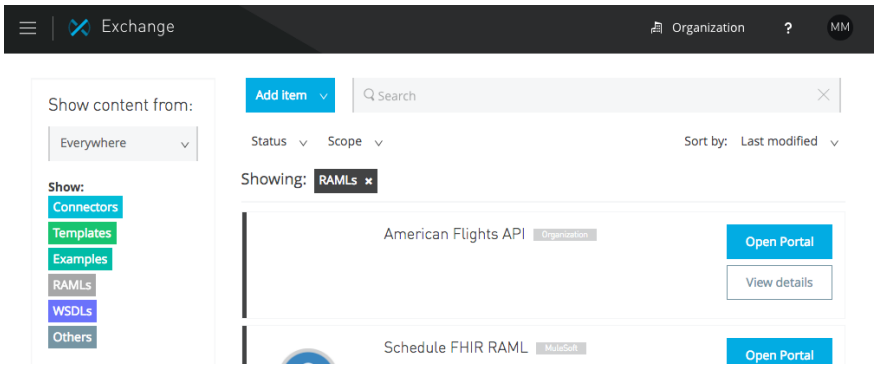
- Anypoint Exchange is a library of connectors, templates, examples RAMLs, WSDLs, and more
- What does it contain?
  - MuleSoft provided public resources
    - You can work with MuleSoft to get any connectors you build certified and added
  - Any private resources added to your organization's private Exchange
    - Everything you add is added to your private Exchange
- How do I get to it?
  - From Anypoint Platform main menu
  - https://www.mulesoft.com/exchange
  - Directly from Anypoint Studio

Anypoint Exchange



Walkthrough 2-5: Add an API to the Anypoint Exchange

- Give yourself permission to publish items on the Exchange
- Add a new RAML API to the private Exchange
- Submit an item for approval
- Approve and publish an item to the private Exchange



Summary

Summary

MuleSoft

- RAML is a non-proprietary, standards-based API description language spec that is simple, succinct, and intuitive to use
  - Data structure hierarchy is specified by indentation, not markup characters
- API Designer can be used to write API definitions with RAML
- Documentation is auto-generated from a RAML file and displayed in an API Console
- API Console can be used to test APIs
- If an API is not yet implemented, the mocking service can be used to test it and return the example data specified in RAML

Summary

MuleSoft

- API portals that have an API Console can be auto-generated for an API in the API Manager
- API Portal Designer is used to add content to, to customize, and to set access to API portals
- An API portal should contain all the content a developer needs to get started with, to test, and to use an API
  - Tutorials, code snippets, examples, API Notebooks
- Make an API discoverable by adding it to the Exchange
  - Everything you add is added to your private Exchange
- Anypoint Exchange is a library of connectors, templates, examples RAMLs, WSDLs, and more

RAML resources

MuleSoft

- RAML definitions can be a lot more complex and sophisticated then what we built here

- Training: http://training.mulesoft.com
  - Anypoint Platform Development: API Design (1 day)
- Website: http://raml.org
  - Documentation
  - Tutorials
  - Full spec
  - Resources

RAML 1.0

MuleSoft

- Released May 2016
- May 2016 releases of Anypoint Platform and Studio 6.0.0 only have limited, early-access support for exploring RAML 1.0
- RAML 1.0 is all about empowering developers with even more extensibility, code reuse options, and flexibility introducing
  - Data types
    - A unified, streamlined, and powerful way to model data everywhere in an API
    - YAML-based so simpler than using JSON or XML schema
  - Annotations, libraries, overlays, extensions, and improved security schemes