Module 9: Reusing Patterns

MuleSoft®

Goal

MuleSoft

```
1    #%RAML 1.0 ResourceType
2    post?:
3      description: Add a new <<resourcePathName | !singularize>>
4      displayName: Add new <<resourcePathName | !singularize>>
5      body:
6        type: <<reso  30      get:
7      responses:          31        description: Retrieve a list of customers
                           32        is:
                           33          - cacheable
                           34          - hasAcceptHeader:
                           35            customErrorDataType: CustomErrorMessage
                           36        responses:
                           37          200:
```

2

Objectives

MuleSoft

- Create and reference resource types patterns for reusability
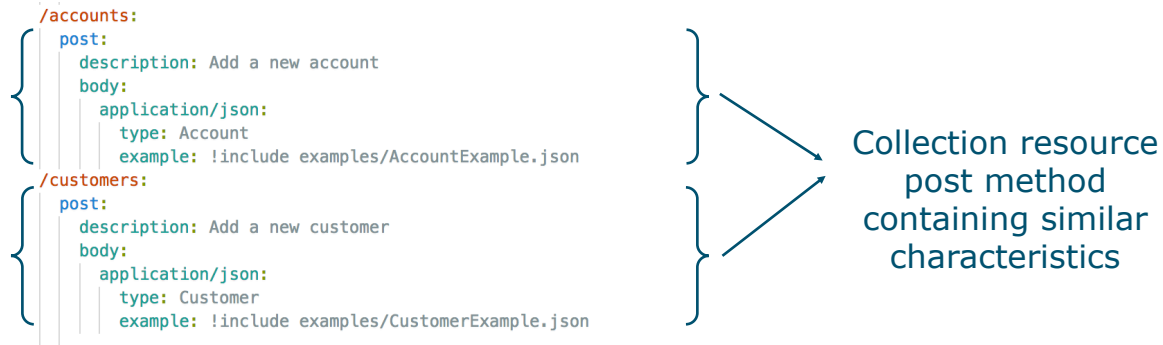- Use traits to modularize methods

3

Using resource types

Introducing resource types

MuleSoft

- Use resource types to modularize common method characteristics in resources
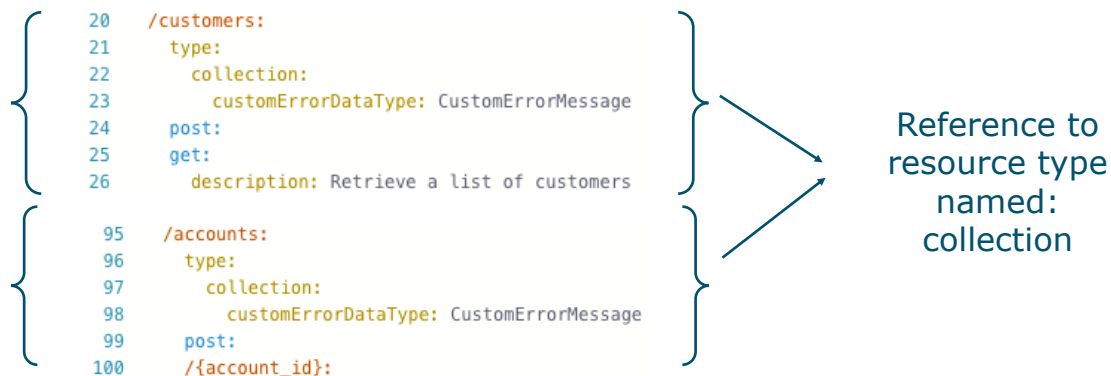- Multiple resource methods can form a resourceType

```
/accounts:
  post:
    description: Add a new account
    body:
      application/json:
        type: Account
        example: !include examples/AccountExample.json
/customers:
  post:
    description: Add a new customer
    body:
      application/json:
        type: Customer
        example: !include examples/CustomerExample.json
```

Collection resource post method containing similar characteristics

All contents © MuleSoft Inc.

Walkthrough 9-1: Define and use a resource type for resources that perform operations on a collection
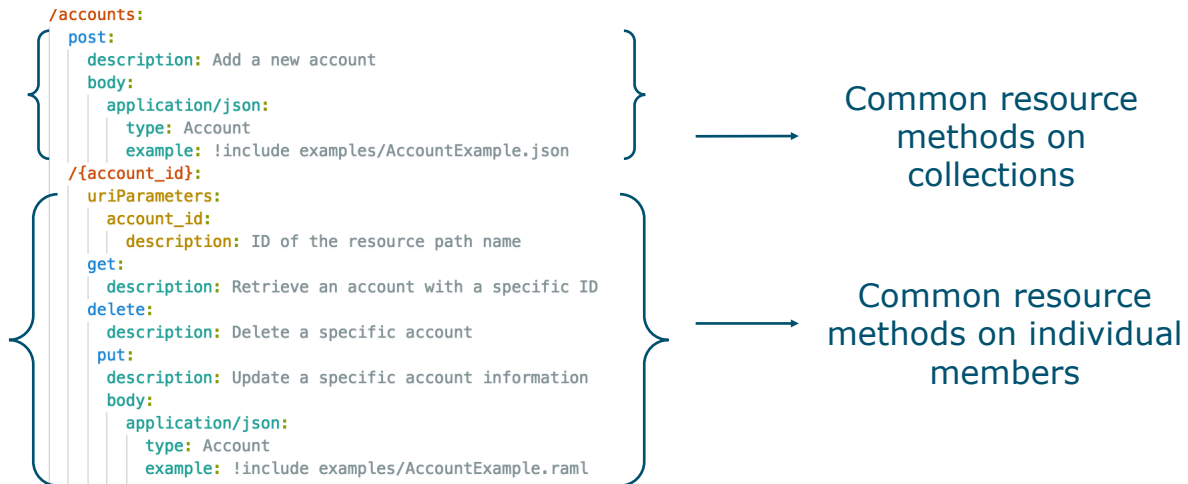
MuleSoft

- Define a collection resource type fragment
- Use a mapping to pass parameter values to the resource type
- Reference the resource type in the RAML API definition

```
20    /customers:
21      type:
22        collection:
23          customErrorDataType: CustomErrorMessage
24      post:
25      get:
26        description: Retrieve a list of customers

95    /accounts:
96      type:
97        collection:
98          customErrorDataType: CustomErrorMessage
99      post:
100     /{account_id}:
```

Reference to resource type named: collection

All contents © MuleSoft Inc.

6

3

Member resource methods have different characteristics compared to collection resources

MuleSoft

```
/accounts:
  post:
    description: Add a new account
    body:
      application/json:
        type: Account
        example: !include examples/AccountExample.json
  /{account_id}:
    uriParameters:
      account_id:
        description: ID of the resource path name
    get:
      description: Retrieve an account with a specific ID
    delete:
      description: Delete a specific account
    put:
      description: Update a specific account information
      body:
        application/json:
          type: Account
          example: !include examples/AccountExample.raml
```

Common resource methods on collections

Common resource methods on individual members

All contents © MuleSoft Inc.

7

Walkthrough 9-2: Define and use a resource type for resources that perform operations on a member

MuleSoft

- Define a member resource type fragment
- Use mappings to pass parameter values to the resource type
- Reference the resource type in the RAML API definition

```
59      /{customer_id}:
60        type:
61          member:
62            exampleValue: !include examples/CustomerExample.raml
63            customErrorDataType: CustomErrorMessage
64        get:
65        patch:
66        delete:
              82      /{account_id}:
              83        type:
              84          member:
              85            exampleValue: !include examples/AccountExample.raml
              86            customErrorDataType: CustomErrorMessage
              87        get:
              88        put:
              89        delete:
```
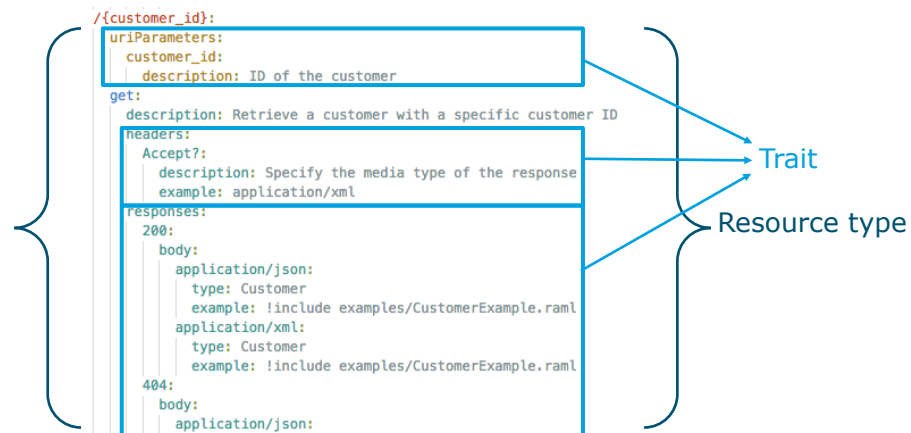
All contents © MuleSoft Inc.

8

Using traits

Introducing traits

MuleSoft

- Smallest reusable component that can define common characteristics across resource methods and resources
  - They can be a part of the resource (query or URI parameters), method or the resource type that the method belongs to

```
/{customer_id}:
  uriParameters:
    customer_id:
      description: ID of the customer
  get:
    description: Retrieve a customer with a specific customer ID
    headers:
      Accept?:
        description: Specify the media type of the response
        example: application/xml
    responses:
      200:
        body:
          application/json:
            type: Customer
            example: !include examples/CustomerExample.raml
          application/xml:
            type: Customer
            example: !include examples/CustomerExample.raml
      404:
        body:
          application/json:
```

Trait

Resource type

10

5

Walkthrough 9-4: Define and use various traits for resources and methods

MuleSoft

- Consume a cacheable trait from Anypoint Exchange
- Define a flexible content type trait to be applied to the resource method with an Accept header
- Refactor the resource methods to use these traits

```
30    get:
31      description: Retrieve a list of customers
32      is:
33        - cacheable
34        - hasAcceptHeader:
35            customErrorDataType: CustomErrorMessag
36      responses:
37        200:
```

/customers : get   Try it

Traits: cacheable, hasAcceptHeader
Retrieve a list of customers

Request

GET *https://mocksvc.mulesoft.com/mocks/ed87b212-aa47-*
     410b-9e19-56f864b5bf09/mocks/5c81644c-33c9-435d
     -96a7-a811a5e969e8/customers

Headers   Hide ∧

Parameter	Type	Description
Accept	string	Specify the media type of the response to be returned

Example value: **application/xml**

All contents © MuleSoft Inc.   11

Summary

Summary

- Resource types allow reusability of method definitions across the entire RAML API definition
- Traits also helps achieve reuse and modularity which allows for easier maintenance and design
  - Traits can be a part of a resource type

13