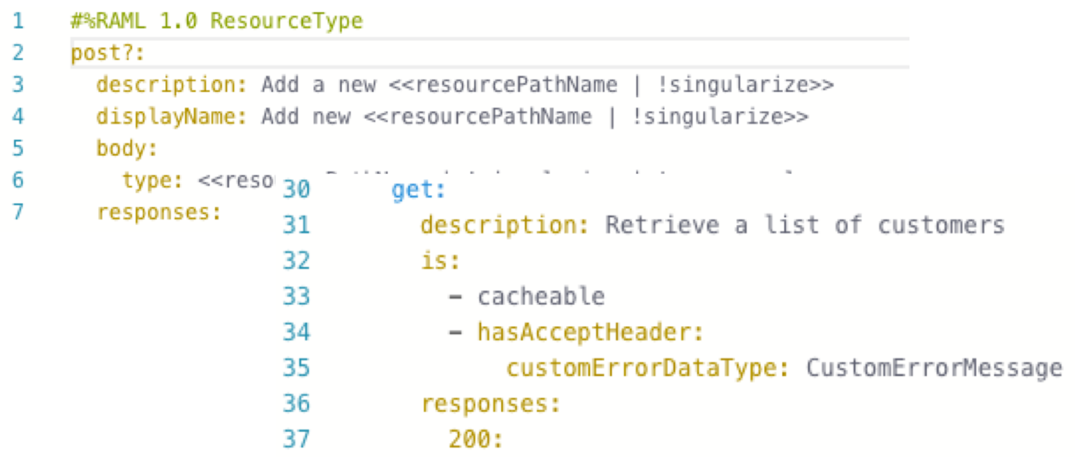
Module 9: Reusing Patterns

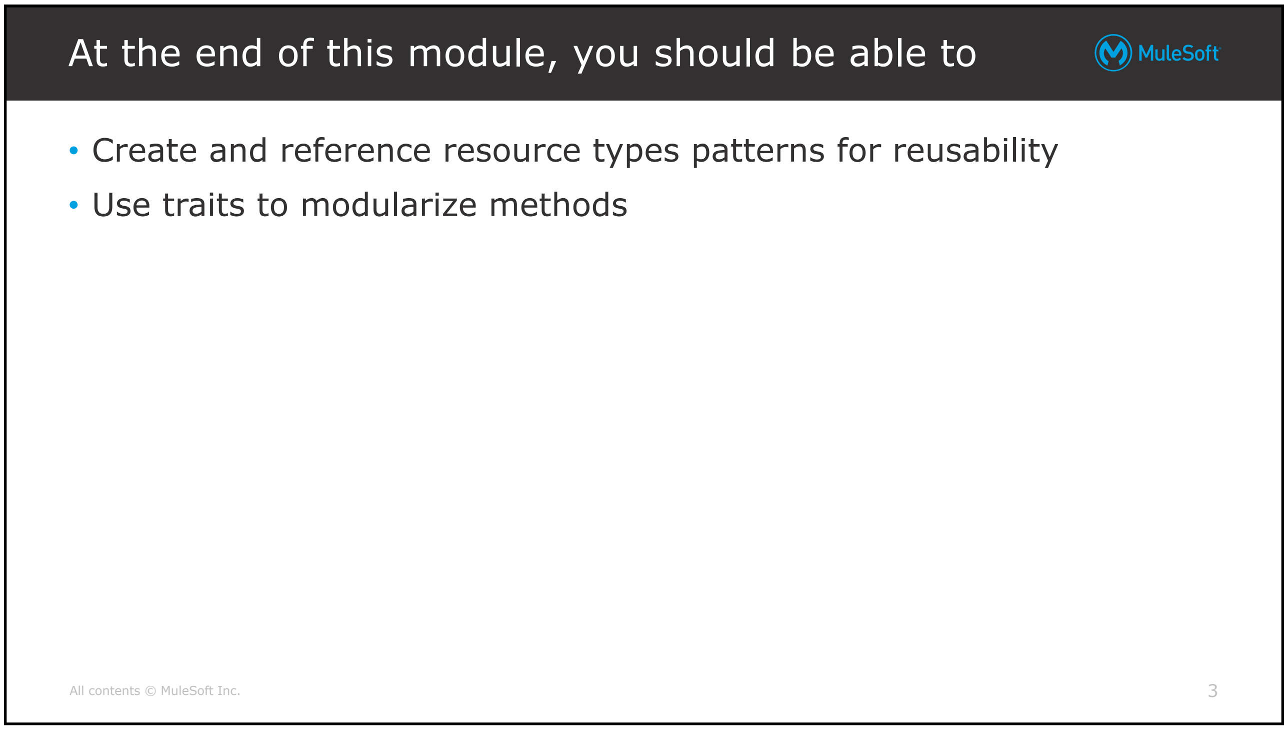
Goal



```
#%RAML 1.0 ResourceType
post?:
  description: Add a new <<resourcePathName | !singularize>>
  displayName: Add new <<resourcePathName | !singularize>>
  body:
    type: <<reso
  responses:
```

```
get:
  description: Retrieve a list of customers
  is:
    - cacheable
    - hasAcceptHeader:
        customErrorDataType: CustomErrorMessage
  responses:
    200:
```

All contents © MuleSoft Inc.



At the end of this module, you should be able to

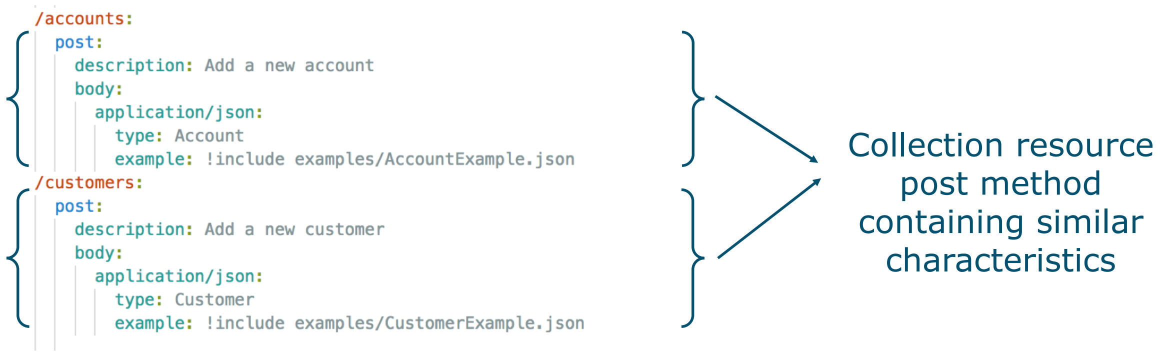
- Create and reference resource types patterns for reusability
- Use traits to modularize methods

All contents © MuleSoft Inc.

Using resource types

Introducing resource types

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



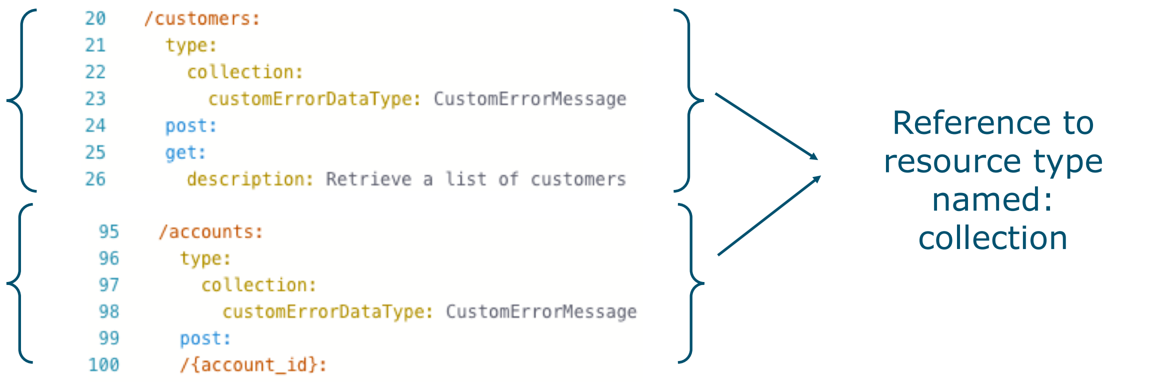
```
/accounts:
  post:
    description: Add a new account
    body:
      application/json:
        type: Account
        example: !include examples/AccountExample.json
/customers:
  post:
    description: Add a new customer
    body:
      application/json:
        type: Customer
        example: !include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

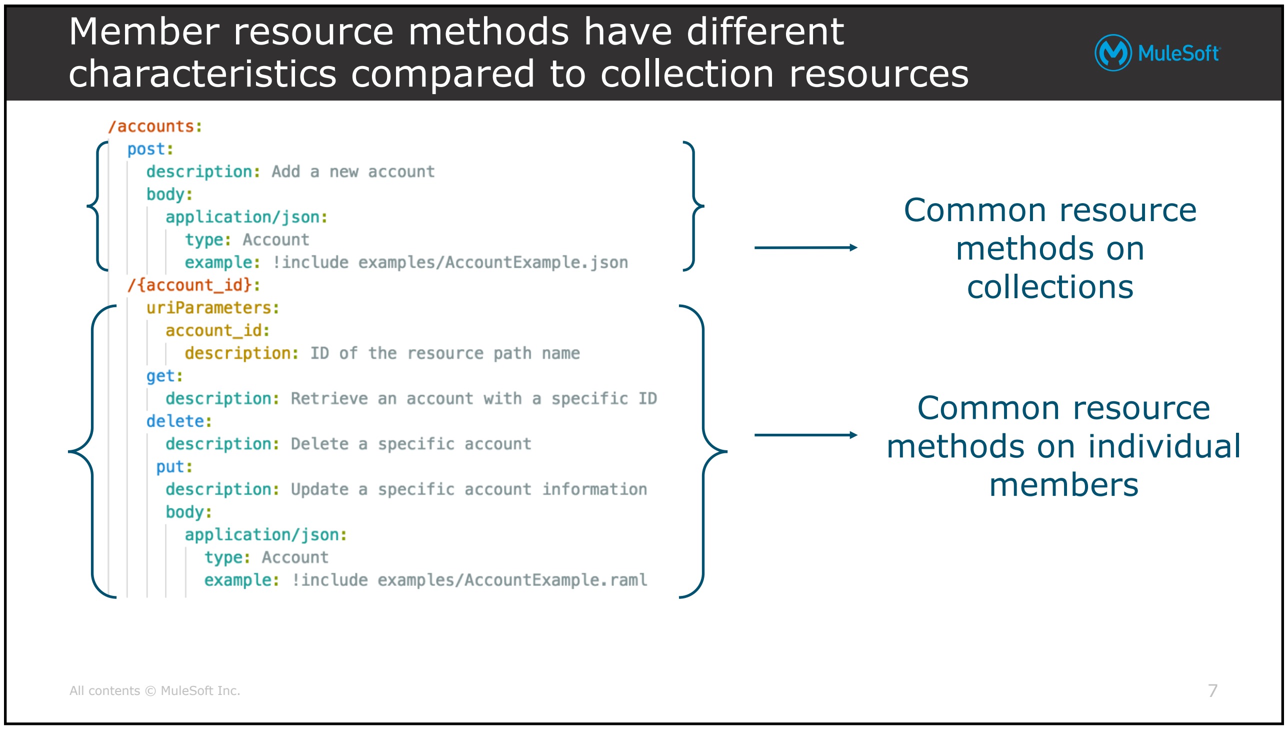
- Define a collection resource type fragment
- Use a mapping to pass parameter values to the resource type
- Reference the resource type in the RAML API definition



```
/customers:
  type:
    collection:
      customErrorDataType: CustomErrorMessage
  post:
  get:
    description: Retrieve a list of customers

/accounts:
  type:
    collection:
      customErrorDataType: CustomErrorMessage
  post:
  /{account_id}:
```

All contents © MuleSoft Inc. 6



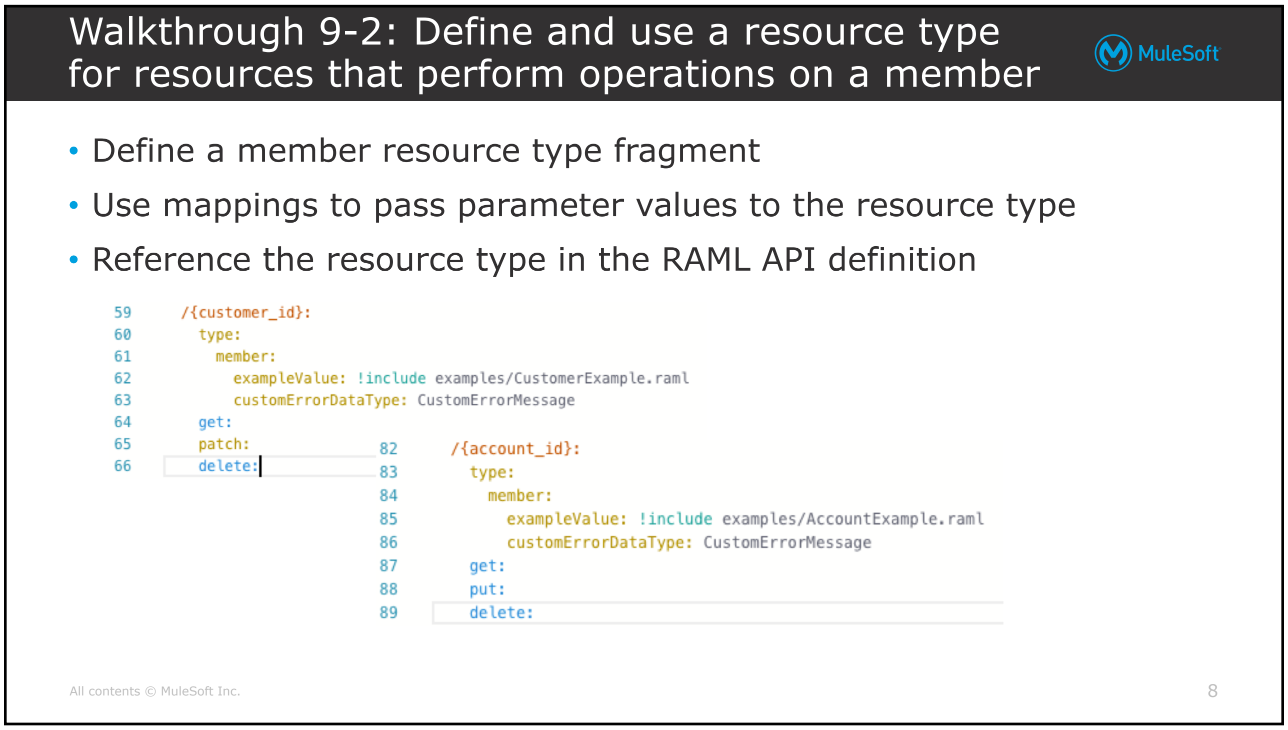
Member resource methods have different characteristics compared to collection resources

```
/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.



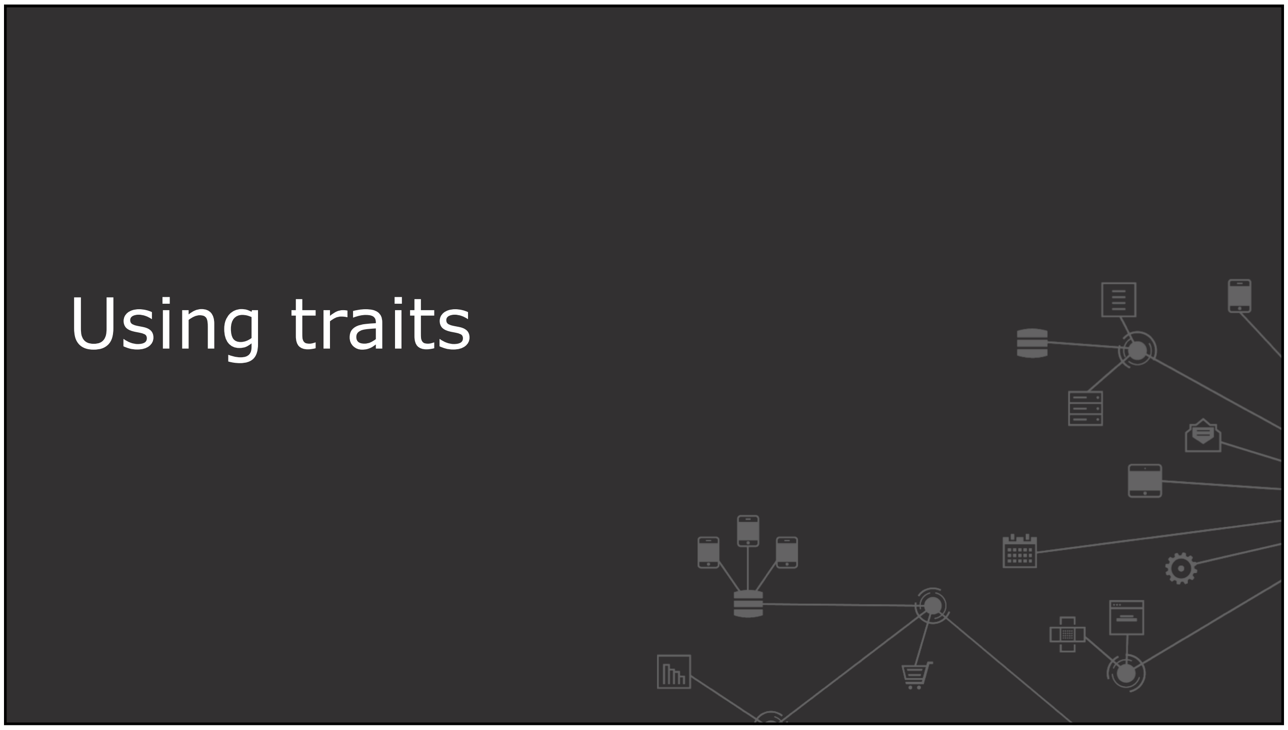
Walkthrough 9-2: Define and use a resource type for resources that perform operations on a member

- 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

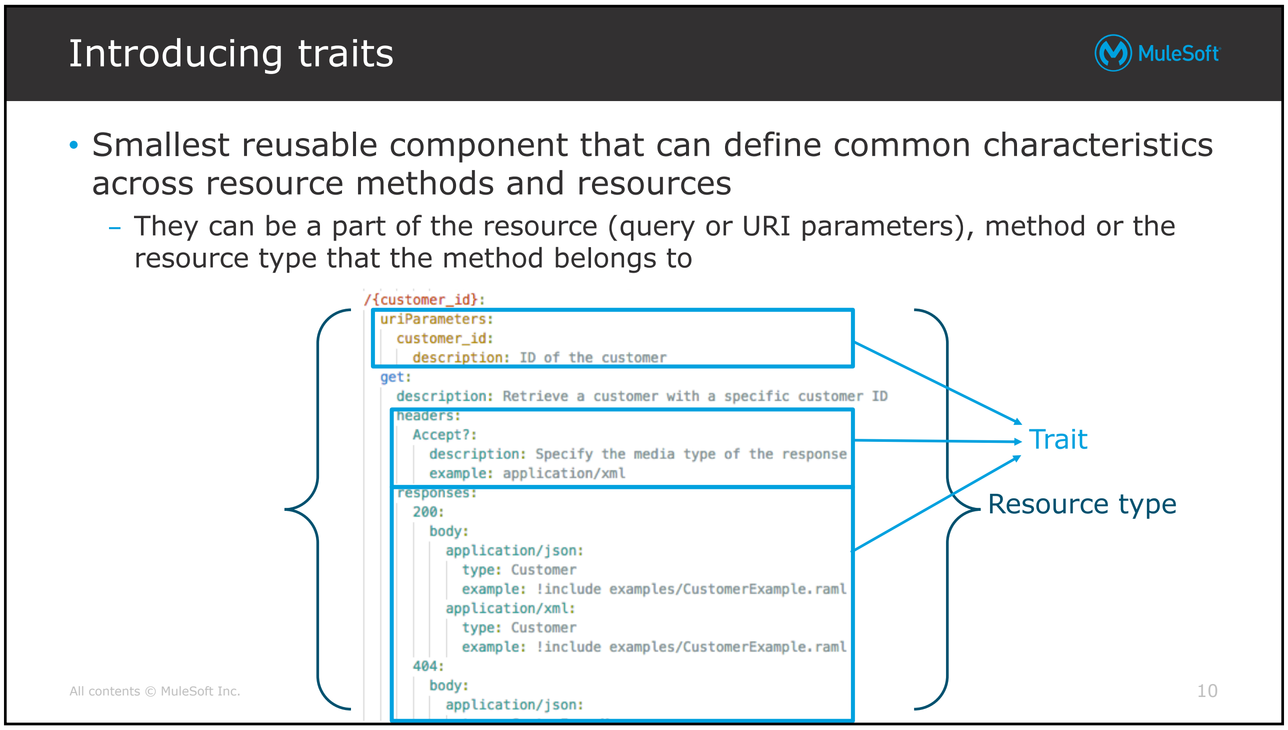
```
  /{customer_id}:
    type:
      member:
        exampleValue: !include examples/CustomerExample.raml
        customErrorDataType: CustomErrorMessage
    get:
    patch:
    delete:
```

```
  /{account_id}:
    type:
      member:
        exampleValue: !include examples/AccountExample.raml
        customErrorDataType: CustomErrorMessage
    get:
    put:
    delete:
```

All contents © MuleSoft Inc.



Using traits



Introducing traits

MuleSoft

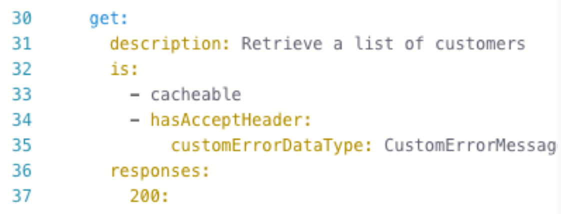
- Smallest reusable component that can define common characteristics across resource methods and resources
  - They can be a part of the resource (query or URI parameters), method or the resource type that the method belongs to

```
/{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:
```

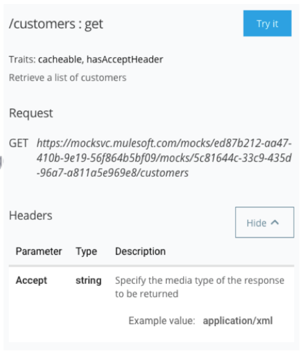
All contents © MuleSoft Inc.

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

- 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



```
  get:
    description: Retrieve a list of customers
    is:
      - cacheable
      - hasAcceptHeader:
          customErrorDataType: CustomErrorMessag
    responses:
      200:
```



/customers : get

Traits: cacheable, hasAcceptHeader

Retrieve a list of customers

Request

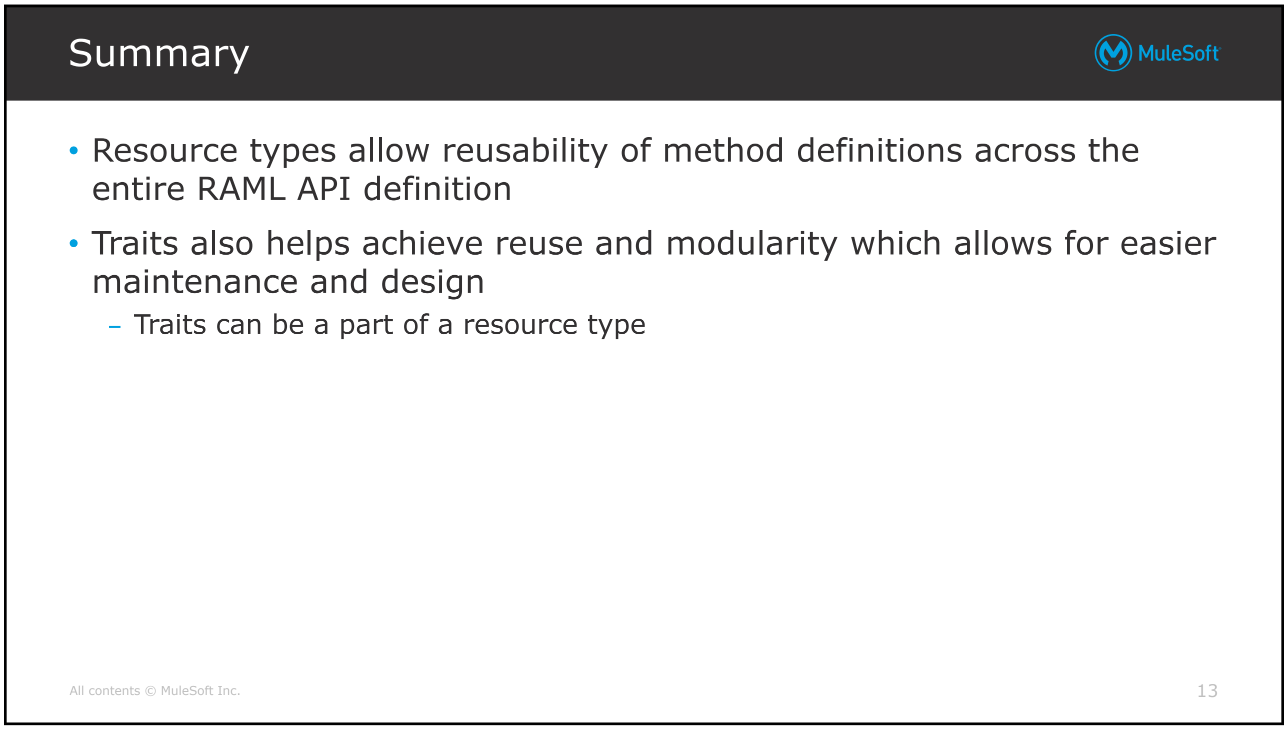
GET *https://mocksvc.mulesoft.com/mocks/ed87b212-aa47-410b-9e19-56f864b5bf09/mocks/5c81644c-33c9-435d-96a7-a811a5e969e8/customers*

Headers

Parameter	Type	Description
Accept	**string**	Specify the media type of the response to be returned Example value: **application/xml**

All contents © MuleSoft Inc.

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

All contents © MuleSoft Inc.