Module 10: Modularizing APIs

Goal

```
#%RAML 1.0 Library
usage: Use the following traits request and response headers and response statu

traits:
  cacheable: !include ../exchange_modules/68ef9520-24e9-4cf2-b2f5-620025690913/
  hasAcceptHeader: !include ../traits/hasAcceptHeader.raml
  hasGetResponse:
    responses:
      200:
        body:
          type: <<resourcePathName | !singul
      404:
        body:
          type: <<customErrorDataType>>
```

```
#%RAML 1.0 Overlay
extends: ../acme-banking-api.raml
usage: Spanish localization

documentation:
  - title: ACME API Bancario Home
    content: |
      **ACME Banca API** permite a los desarrolladores crear aplicaciones que hacen uso de la i

      Esta API contiene una funcionalidad que permite a los desarrolladores recuperar y manipul
  - title: Banco ACME Headline
    content: |
                                                    al de servicios bancarios_ y financieros de la organiza
```

```
#%RAML 1.0 Extension
extends: ../acme-banking-api.raml
baseUri: <<insert prod specific implementation URL>>
```

Objectives

- Use libraries for greater API composability
- Override resource information using overlays
- Use overlays to internationalize resources
- Use extensions to enhance resources

Introducing libraries

Achieving modularity using libraries

MuleSoft

- RAML libraries combine collections of datatypes, resource types, or traits declarations into reusable groups
  - Used mainly to externalize common declarations
  - Can be defined inline like any other fragment file
  - They can also reference fragment files in them
- Libraries can be applied with the **uses** node at the root of RAML API definition
  - The assets in the library are referenced within the document using dot notation For example: library_name.asset_name

```
uses:
  file-type: file-type.raml
traits:
  drm:
    headers:
      drm-key:
resourceTypes:
  file:
    get:
      is: [ drm ]
      responses:
        201:
          body:
            application/json:
              type: file-type.File
```


Walkthrough 10-1: Create and use a library of traits

MuleSoft

- Create a library fragment file
- Refactor and move existing traits inside the library file
- Create and define new resource method request and response traits
- Refactor the API definition to reuse the traits from the library

```
#%RAML 1.0 Library
usage: Use the following traits request and response headers and response statu

traits:
  cacheable: !include ../exchange_modules/68ef9520-24e9-4cf2-b2f5-620025690913/
  hasAcceptHeader: !include ../traits/hasAcceptHeader.raml
  hasGetResponse:
    responses:
      200:
        body:
          type: <<resourcePathName | !singu
      404:
        body:
          type: <<customErrorDataType>>
```

```
    is:
      - Traits.cacheable
      - Traits.hasAcceptHeader:
          customErrorDataType: CustomErrorMessage
    responses:
```

Introducing overlays

Overriding some elements of RAML API definition using overlays

MuleSoft

- Overrides nodes of a RAML API definition but preserves its behavioral, functional aspects
  - Resources, methods, parameters, bodies, responses are some behavioral nodes that cannot be overridden in overlays
- Most commonly used for
  - Adding hooks to testing and monitoring tools
  - Appending metadata relevant to APIs
  - Providing translated human documentation

Walkthrough 10-2: Internationalize documentation and resource descriptions using an overlay

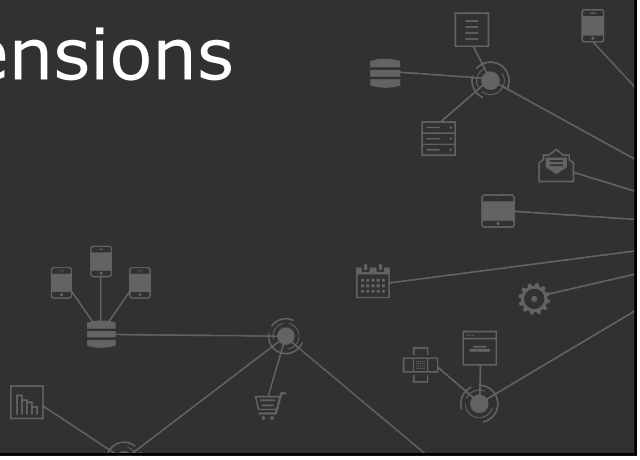
- Create a RAML overlay fragment file
- Add documentation and resource method description nodes in the overlay file to support Spanish localization

```
1  #%RAML 1.0 Overlay
2  extends: ../acme-banking-api.raml
3  usage: Spanish localization
4
5  documentation:
6    - title: ACME API Bancario Home
7      content: |
8        **ACME Banca API** permite a los desarrolladores crear aplicaciones que h
9
10       Esta API contiene una funcionalidad que permite a los desarrolladores rec
11   - title: Banco ACME Headline
12     content: |
13       **Banco ACME** es una _multinacional de servicios bancarios_ y financiero
14
```

ACME Banca API permite a los desarrolladores crear aplicaciones que hacen uso de la información del recurso a los métodos implementados en la API.

Esta API contiene una funcionalidad que permite a los desarrolladores recuperar y manipular el *cliente*, la *cuenta* y la información de la *transacción*. Echa un vistazo a la API Portal para más detalles.

Introducing extensions

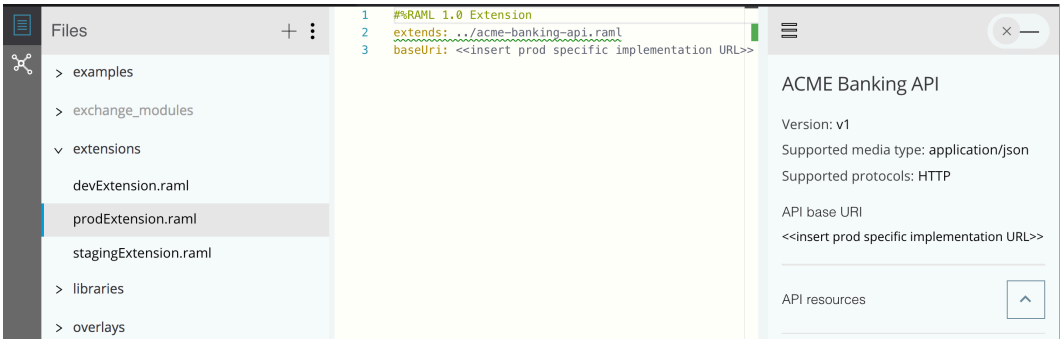


Extending behavior of RAML elements using extensions

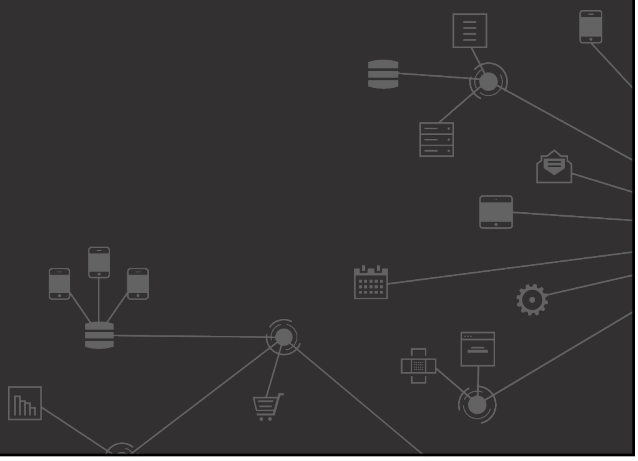
- Extensions broaden an API definition by adding to the behavior and the functional aspects
- Used to create variants of the API, targeted to specific classes of API consumers
  - For example: A certain class of audience need to have ability to work with methods that post data to the system or delete from them (admin privileges)
  - Also used to add environment specific service URL

Walkthrough 10-3: Define and use API extensions to promote portability to test in multiple environments

- Create API extension files
- Add a baseUri parameter to the extension files and enter a placeholder for environment specific implementation URL



Summary



Summary

MuleSoft

- RAML supports modular design
- Use libraries to reuse them in API definitions
- Use overlays to override non-functional aspects like testing tools, metadata information, support for internationalization
- Use extensions to extend functionality of the API
  - Useful in places where functionality is to be divided based of the relevance of the audience