TOC

LAB : Traits

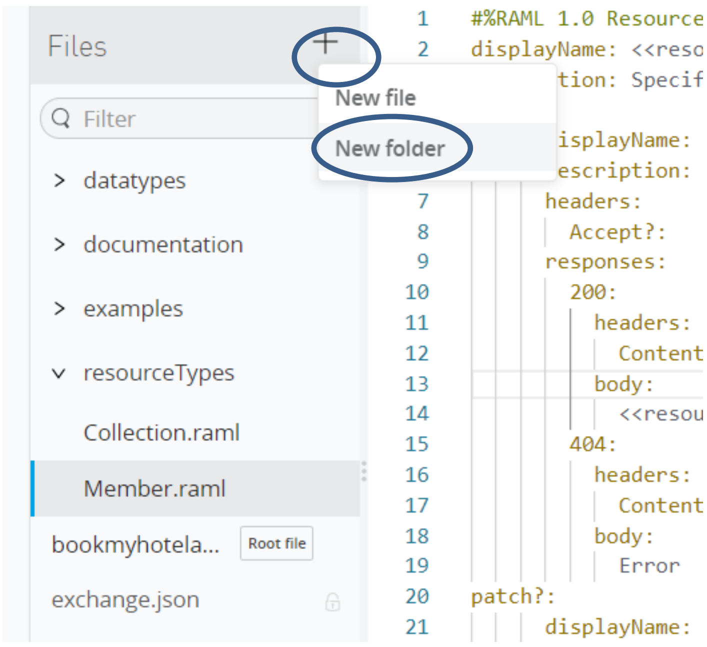
In this lab you will -

a) Create Traits for Accept Header and Error response

b) Use these traits in ResourceType Fragments

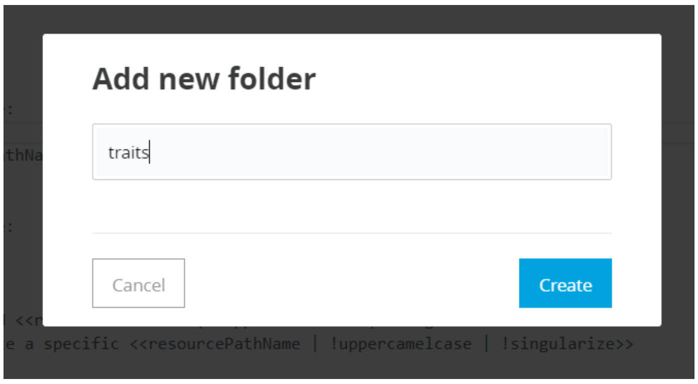
Step 1 – Create Trait Fragment for Accept Header

1. Click on + sign and then Create Folder from the Menu as shown below :-

a. 

```
#%RAML 1.0 Resource
displayName: <<reso
tion: Specif

isplayName:
escription:
    headers:
      Accept?:
    responses:
      200:
        headers:
          Content
        body:
          <<resou
      404:
        headers:
          Content
        body:
          Error
patch?:
    displayName:
```

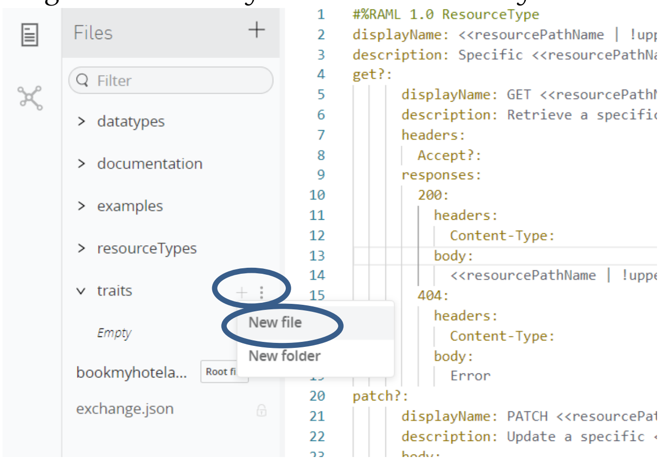
2. In the dialogue box, specify the name of the folder as **traits** and then click on Create button



```
e a specific <<resourcePathName | !uppercamelcase | !singularize>>
```

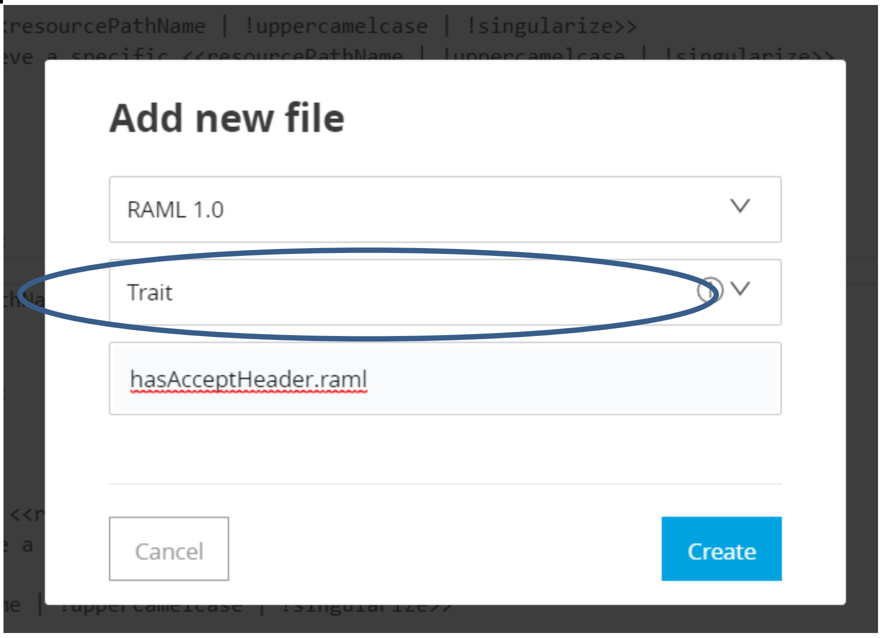
3. In the left hand side you can see the traits folder created and it is empty as of now

4. Click on + sign near the traits folder and click on new file

a. 

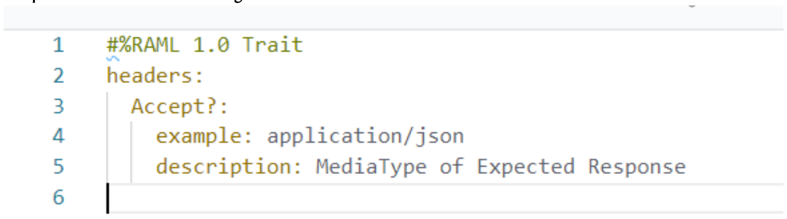
```
#%RAML 1.0 ResourceType
displayName: <<resourcePathName | !upp
description: Specific <<resourcePathNa
get?:
    displayName: GET <<resourcePathN
    description: Retrieve a specific
    headers:
      Accept?:
    responses:
      200:
        headers:
          Content-Type:
        body:
          <<resourcePathName | !uppe
      404:
        headers:
          Content-Type:
        body:
          Error
patch?:
    displayName: PATCH <<resourcePa
    description: Update a specific
```

5. In the Add New File dialogue box, select **Trait** from the drop down and specify the name as **hasAcceptHeader.raml**

   a. 

   b. This will create a Trait fragment – hasAcceptHeader.raml

6. In hasAcceptHeader.raml, define as shown below:-



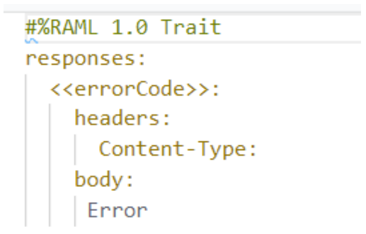
```
#%RAML 1.0 Trait
headers:
  Accept?:
    example: application/json
    description: MediaType of Expected Response

```

   a.

Congratulations, you have completed this step!!

Step 2 – Create Trait Fragment for Error Response

1. Click on + sign near the traits folder and click on new file
2. In the Add New File dialogue box, select **Trait** from the drop down and specify the name as **hasError.raml.** This will create a Trait fragment – hasError.raml
3. In hasError.raml, define as shown below:-



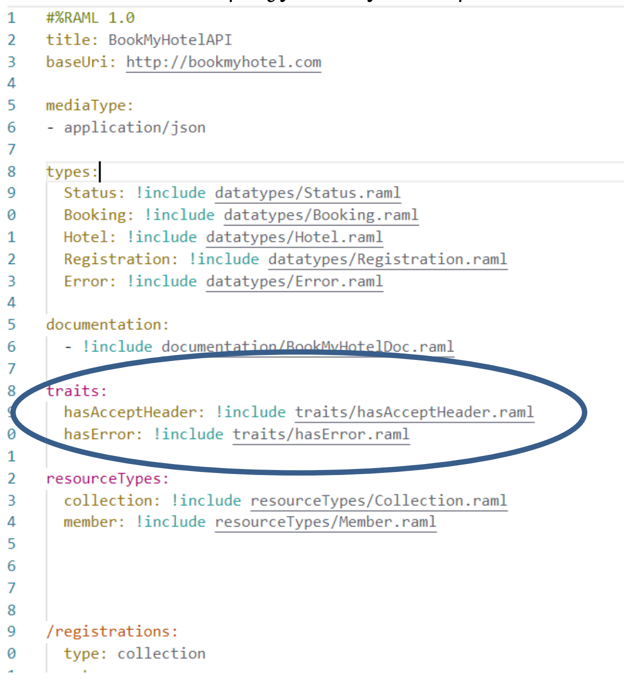
```
#%RAML 1.0 Trait
responses:
  <<errorCode>>:
    headers:
      Content-Type:
    body:
     Error
```

a.
b. Here errorCode is the user defined parameter and we have to pass value while using the trait

Congratulations, you have completed this step!!

Step 3 – Specify traits in root RAML file

1. Open Root RAML file
2. Define the **traits** node and specify the key/value pair as shown below :-



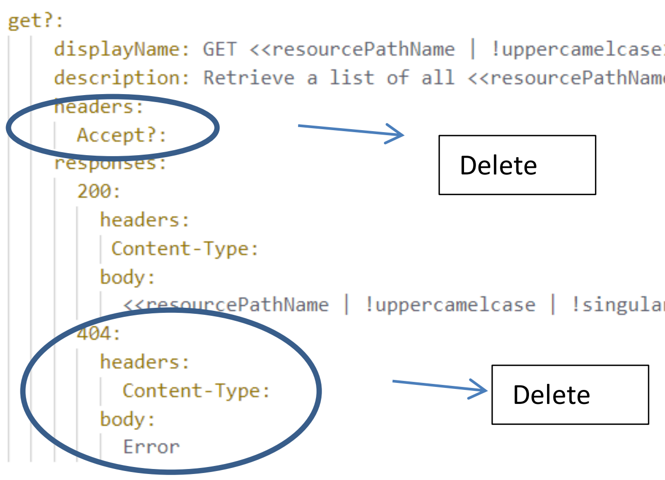
```
#%RAML 1.0
title: BookMyHotelAPI
baseUri: http://bookmyhotel.com

mediaType:
- application/json

types:
  Status: !include datatypes/Status.raml
  Booking: !include datatypes/Booking.raml
  Hotel: !include datatypes/Hotel.raml
  Registration: !include datatypes/Registration.raml
  Error: !include datatypes/Error.raml

documentation:
  - !include documentation/BookMyHotelDoc.raml

traits:
  hasAcceptHeader: !include traits/hasAcceptHeader.raml
  hasError: !include traits/hasError.raml

resourceTypes:
  collection: !include resourceTypes/Collection.raml
  member: !include resourceTypes/Member.raml




/registrations:
  type: collection
```

a.
b. The trait names are specified as hasAcceptHeader and hasError
c. As the value , the trait fragments are included

Congratulations, you have completed this step!!

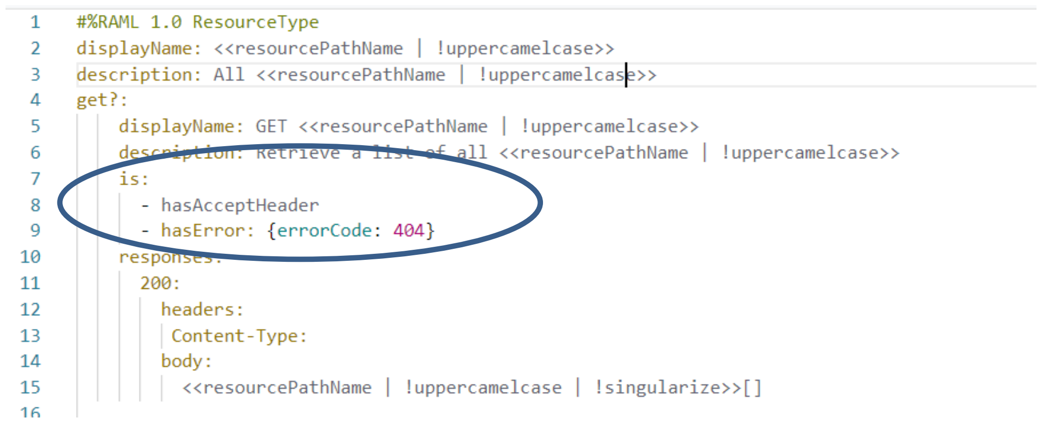
Step 4 – Use Traits in Collection ResourceType Fragment

1. Open Collection ResourceType fragment - Collection.raml
2. From the get method , remove Accept header section and 404 error response section highlighted below :-



```
get?:
  displayName: GET <<resourcePathName | !uppercamelcase>>
  description: Retrieve a list of all <<resourcePathName | !uppercamelcase>>
  headers:
    Accept?:
  responses:
    200:
      headers:
        Content-Type:
      body:
        <<resourcePathName | !uppercamelcase | !singularize>>[]
    404:
      headers:
        Content-Type:
      body:
        Error
```

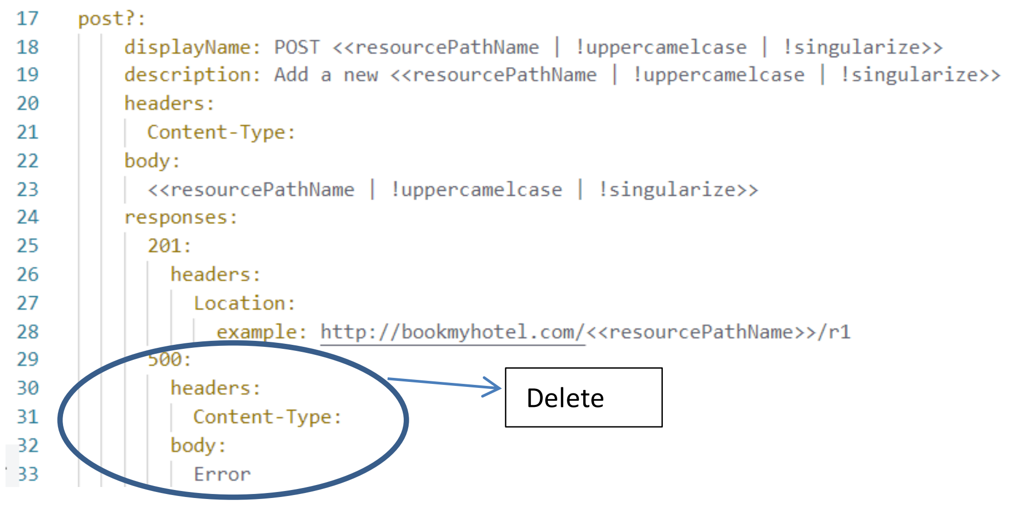
3.
4. In get method, use traits as shown below:-



```
#%RAML 1.0 ResourceType
displayName: <<resourcePathName | !uppercamelcase>>
description: All <<resourcePathName | !uppercamelcase>>
get?:
  displayName: GET <<resourcePathName | !uppercamelcase>>
  description: Retrieve a list of all <<resourcePathName | !uppercamelcase>>
  is:
    - hasAcceptHeader
    - hasError: {errorCode: 404}
  responses:
    200:
      headers:
        Content-Type:
      body:
        <<resourcePathName | !uppercamelcase | !singularize>>[]
```

   a.

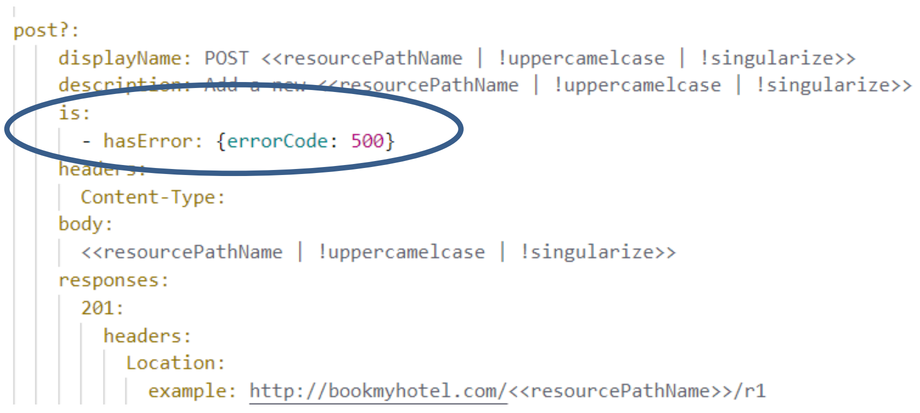
5. From the post method , remove 500 error response section highlighted below :-



```
post?:
  displayName: POST <<resourcePathName | !uppercamelcase | !singularize>>
  description: Add a new <<resourcePathName | !uppercamelcase | !singularize>>
  headers:
    Content-Type:
  body:
    <<resourcePathName | !uppercamelcase | !singularize>>
  responses:
    201:
      headers:
        Location:
          example: http://bookmyhotel.com/<<resourcePathName>>/r1
    500:
      headers:
        Content-Type:
      body:
        Error
```

   a.

6. In post method, use trait as shown below:-

   a. 

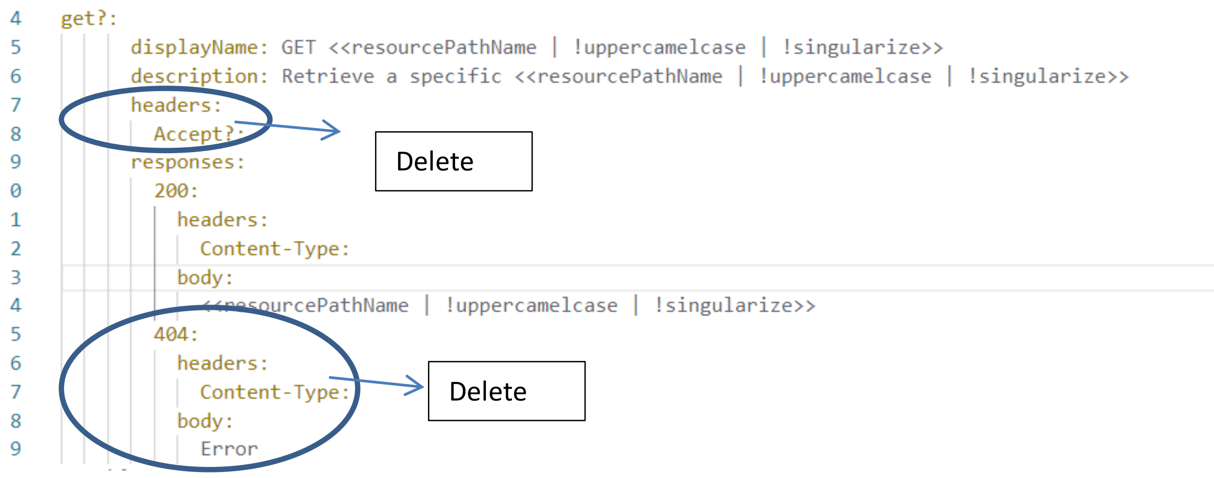
```
post?:
    displayName: POST <<resourcePathName | !uppercamelcase | !singularize>>
    description: Add a new <<resourcePathName | !uppercamelcase | !singularize>>
    is:
      - hasError: {errorCode: 500}
    headers:
      Content-Type:
    body:
      <<resourcePathName | !uppercamelcase | !singularize>>
    responses:
      201:
        headers:
          Location:
            example: http://bookmyhotel.com/<<resourcePathName>>/r1
```

Congratulations, you have completed this step!!

Step 5 – Use Traits in Member ResourceType Fragment

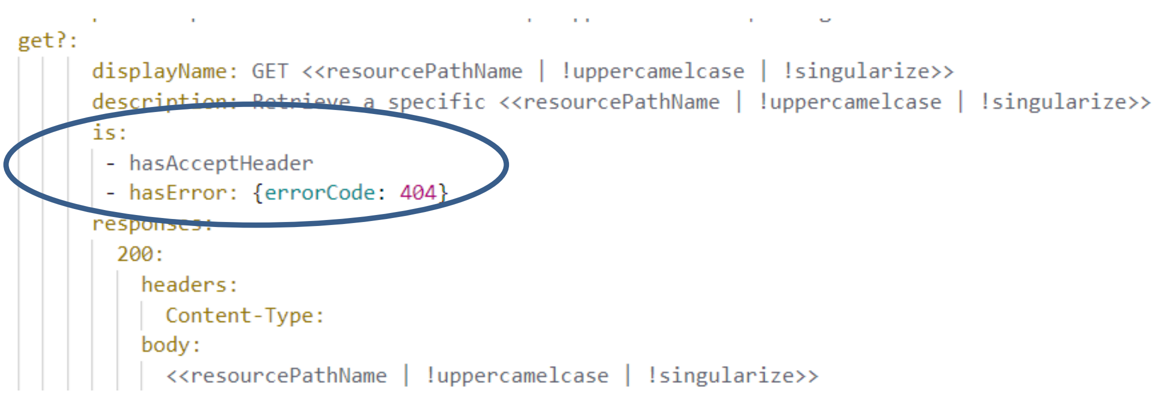
1. Open Member ResourceType fragment - Member.raml

2. From the get method , remove Accept header section and 404 error response section highlighted below :-

   a. 

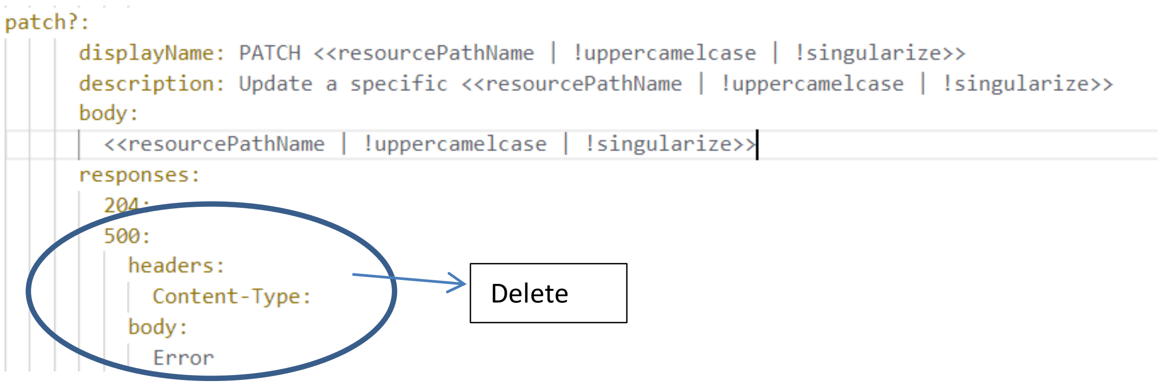
```
get?:
      displayName: GET <<resourcePathName | !uppercamelcase | !singularize>>
      description: Retrieve a specific <<resourcePathName | !uppercamelcase | !singularize>>
      headers:
        Accept?:
      responses:
        200:
          headers:
            Content-Type:
          body:
            <<resourcePathName | !uppercamelcase | !singularize>>
        404:
          headers:
            Content-Type:
          body:
            Error
```

3. In get method, use traits as shown below:-

a. 

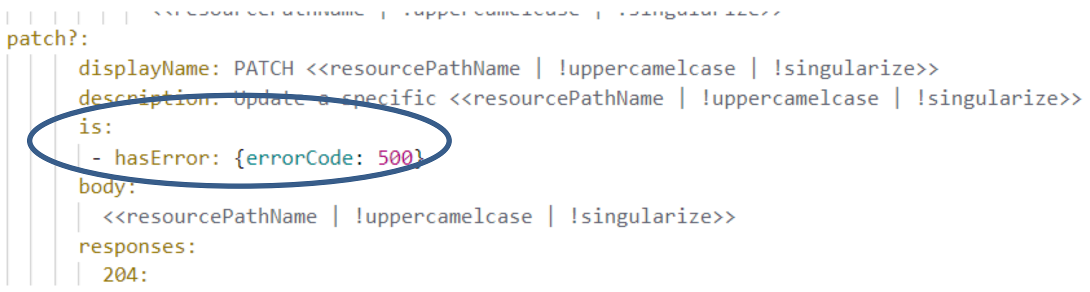
```
get?:
    displayName: GET <<resourcePathName | !uppercamelcase | !singularize>>
    description: Retrieve a specific <<resourcePathName | !uppercamelcase | !singularize>>
    is:
     - hasAcceptHeader
     - hasError: {errorCode: 404}
    responses:
      200:
        headers:
          Content-Type:
        body:
          <<resourcePathName | !uppercamelcase | !singularize>>
```

4. *From the patch method , remove 500 error response section highlighted below :-*

a. 

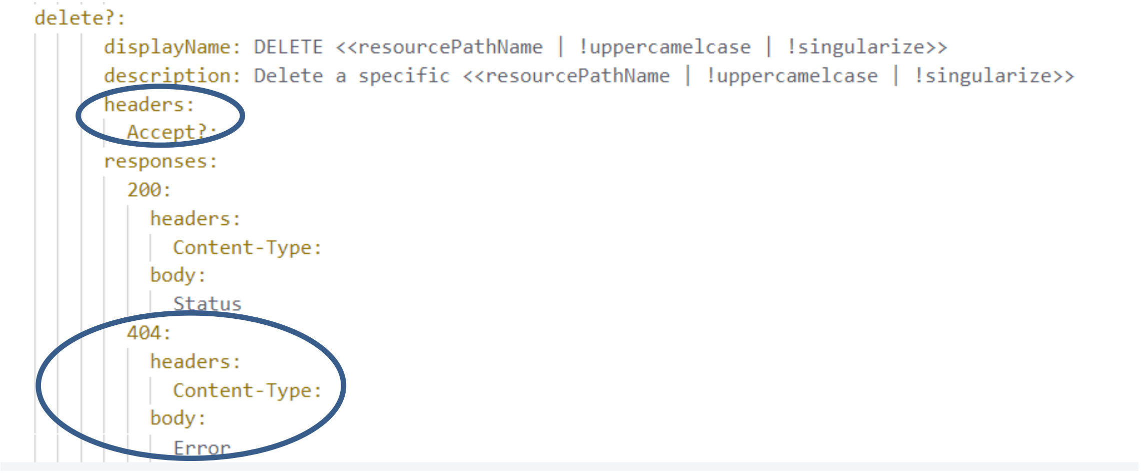
```
patch?:
    displayName: PATCH <<resourcePathName | !uppercamelcase | !singularize>>
    description: Update a specific <<resourcePathName | !uppercamelcase | !singularize>>
    body:
      <<resourcePathName | !uppercamelcase | !singularize>>
    responses:
      204:
      500:
        headers:
          Content-Type:
        body:
          Error
```

5. *In patch method, use trait as shown below:-*

a. 

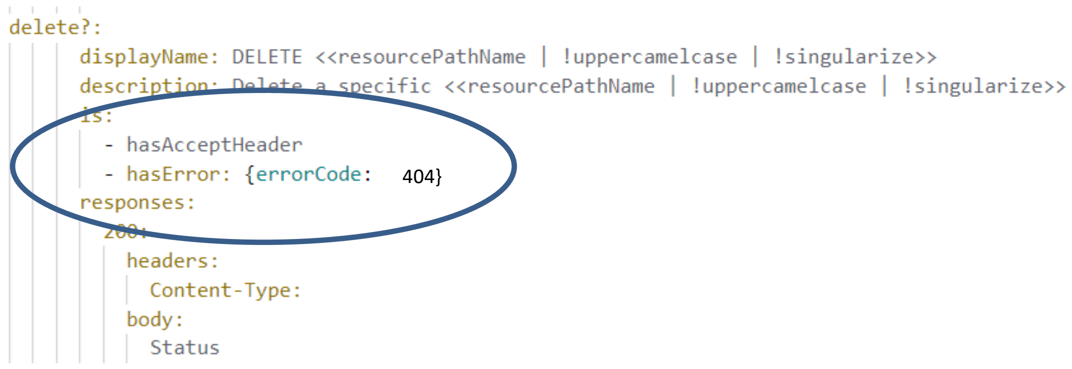
```
patch?:
    displayName: PATCH <<resourcePathName | !uppercamelcase | !singularize>>
    description: Update a specific <<resourcePathName | !uppercamelcase | !singularize>>
    is:
     - hasError: {errorCode: 500}
    body:
      <<resourcePathName | !uppercamelcase | !singularize>>
    responses:
      204:
```

6. *From the delete method , remove Accept header section and 404 error response section highlighted below :-*

a. 

```
delete?:
    displayName: DELETE <<resourcePathName | !uppercamelcase | !singularize>>
    description: Delete a specific <<resourcePathName | !uppercamelcase | !singularize>>
    headers:
      Accept?:
    responses:
      200:
        headers:
          Content-Type:
        body:
          Status
      404:
        headers:
          Content-Type:
        body:
          Error
```

7. In delete method, use traits as shown below:-

   a. 

```
delete?:
    displayName: DELETE <<resourcePathName | !uppercamelcase | !singularize>>
    description: Delete a specific <<resourcePathName | !uppercamelcase | !singularize>>
    is:
      - hasAcceptHeader
      - hasError: {errorCode:   404}
    responses:
      200:
        headers:
          Content-Type:
        body:
          Status
```

8. Go to Root RAML and check in API Console if error codes and accept headers are visible as defined in different methods

Congratulations, you have completed this step!!

Congratulations!! You have completed this exercise.