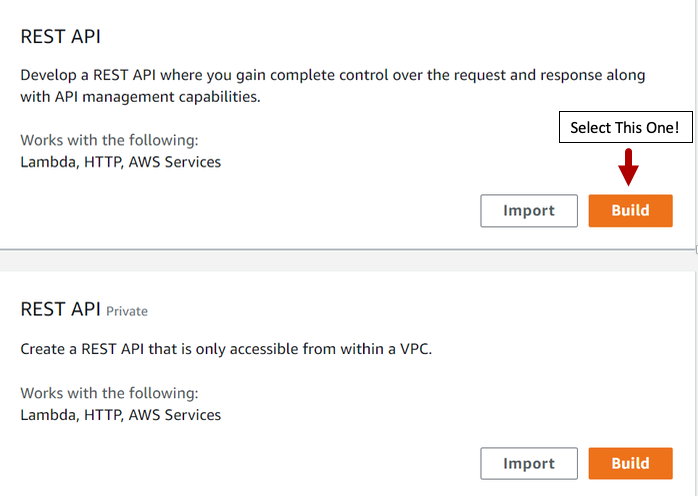
# **CS 470 Project One Guide**

## APIs to Lambda

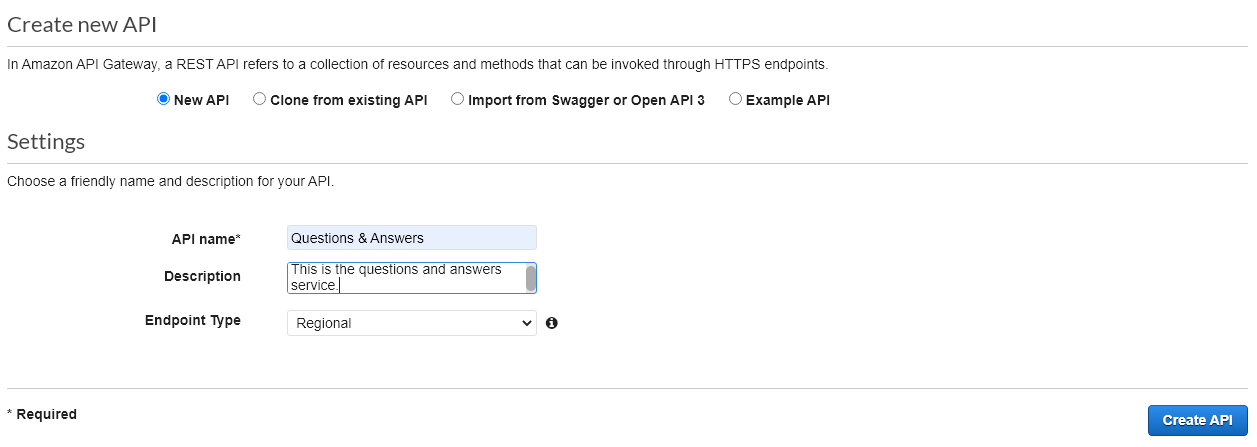
* Please be careful with the case of the API resources. REST is a case-sensitive protocol, and the wrong case will prevent your application from working correctly.
* After every mapping of an API to a Lambda, you will be asked to OK the addition to the permission of the Lambda. Click OK. This allows each individual API you map to a Lambda to have access to that specific Lambda.

### Part One – The Questions API

1. Navigate to the **API Gateway** in the AWS Console. See Module Four Assignment Two for a refresher.
2. Click the orange **Create API** button, then click **Build a REST API**.
   * Do not click the REST API marked “private”.



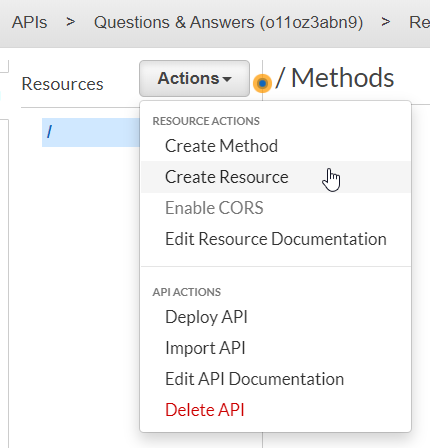
1. Your settings should be **REST**, **New API**, and an **API name** of “Questions & Answers”.



1. Click the **Create API** button.

### Questions Resource

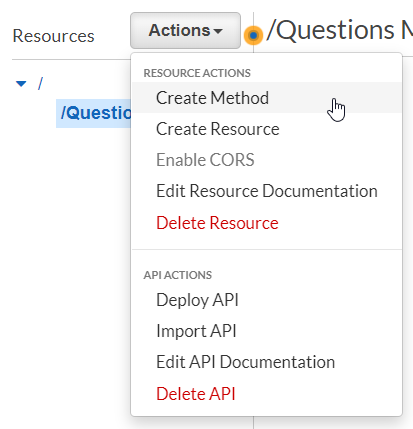
1. Click **Actions** drop-down menu and select **Create Resource**.



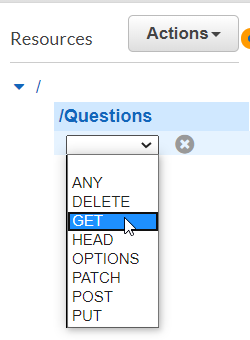
1. Create a resource named “Questions”, and make sure the resource path has a capitalized “Q”.
2. Leave the check boxes un-checked and click **Create Resource**.



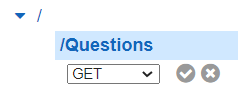
1. With **/Questions** selected, click **Create Method** in the **Actions** drop-down menu.



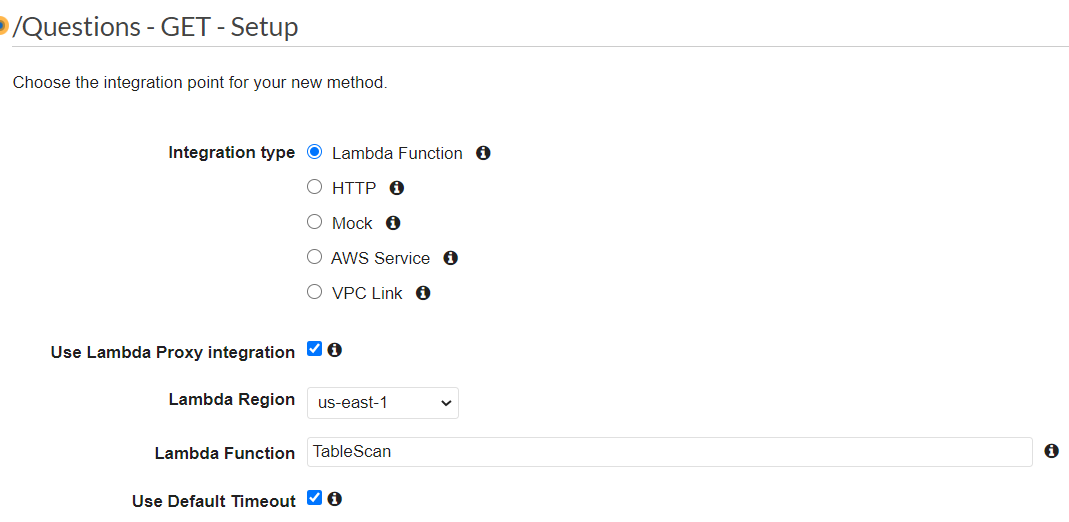
1. Select the GET method in the new drop-down box that is created.



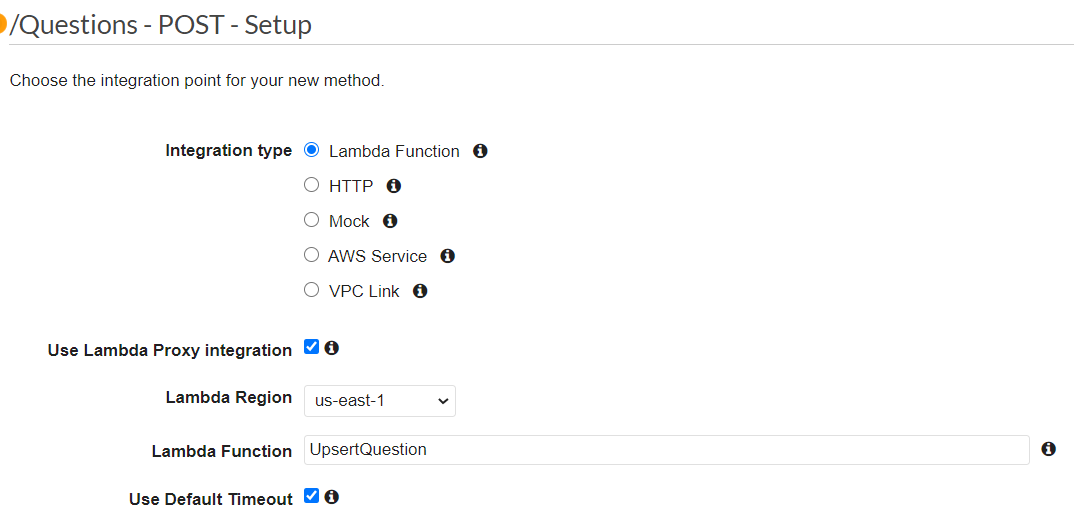
1. Click the checkmark symbol that appears between the method drop-down menu and the X.



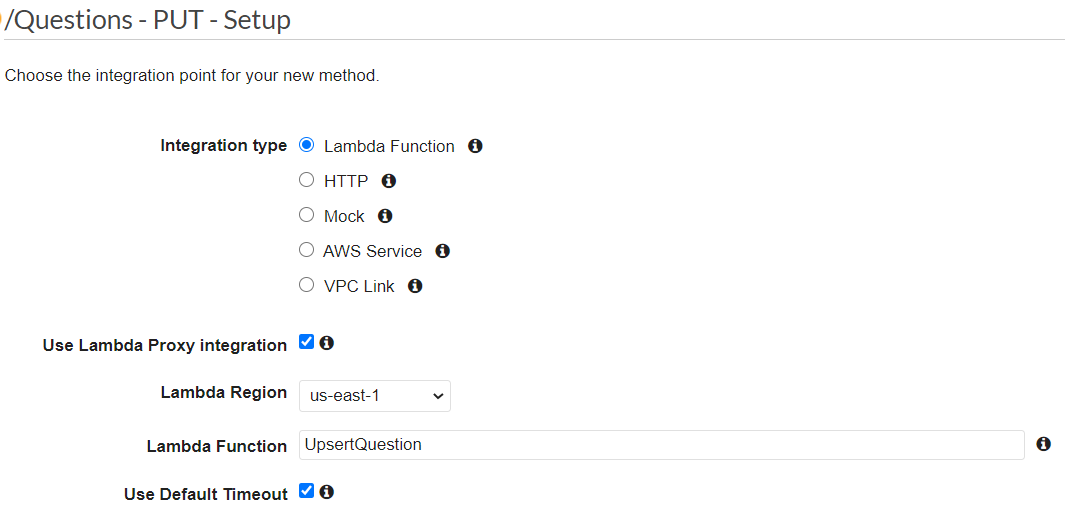
1. Set up your GET method as a Lambda Function, using Lambda Proxy Integration, in region **us-east-1** and with the Lambda Function **TableScan**.



1. Click the **Save** button and then click the **OK** button when asked about **Adding Permission to Lambda Function**.
2. Add the POST method to “Questions” with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **UpsertQuestion**.

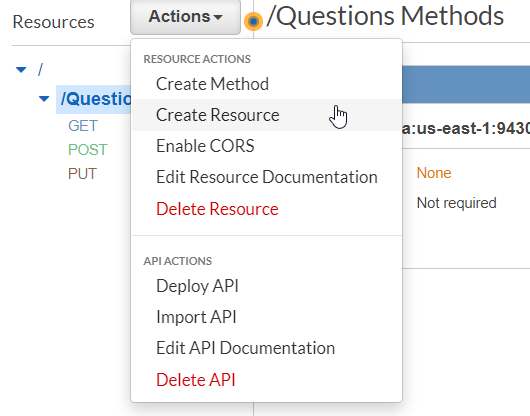


1. Add the PUT method to “Questions” with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **UpsertQuestion**.



### The {id} Sub-Resource

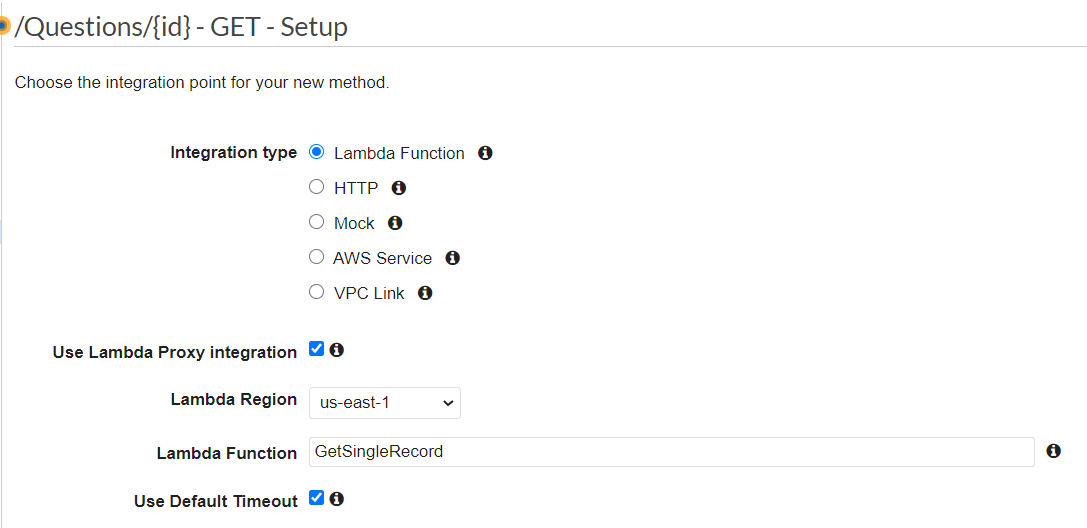
1. Select the **/Questions** resource and click **Create Resource** under the **Actions** menu.



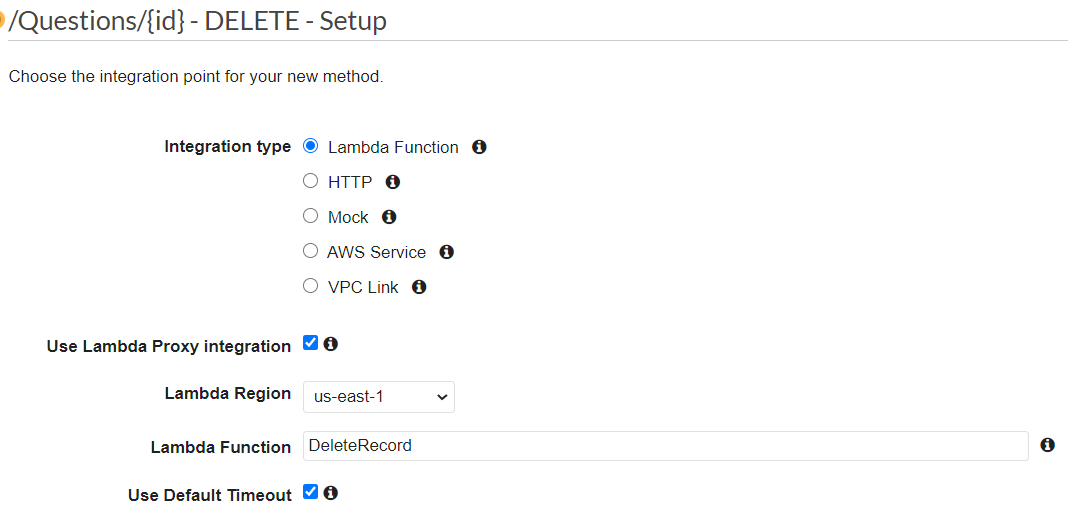
1. Create a resource named **{id}**. Pay special attention to the resource path – it should be **/Questions/{id}**. Leave both check boxes unchecked.



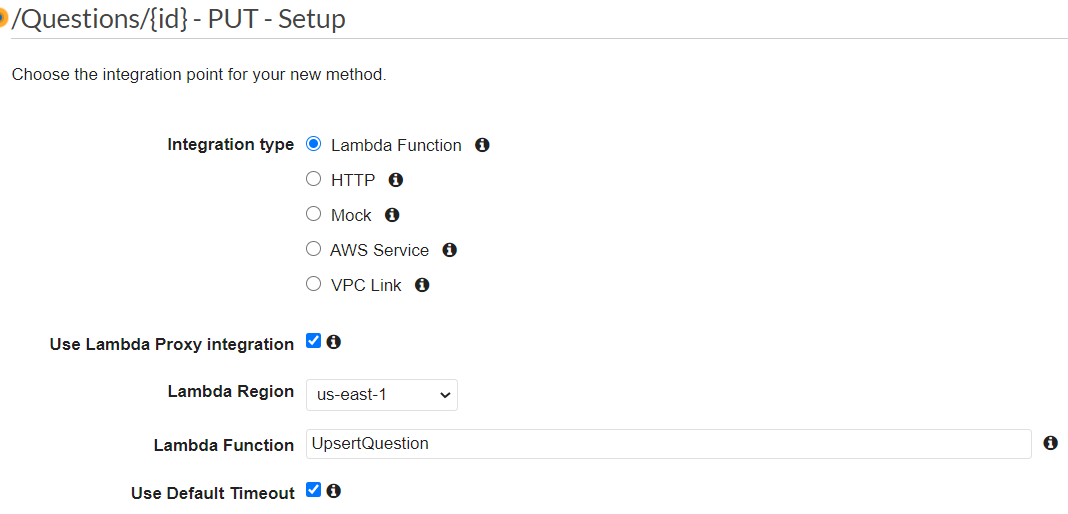
1. Click the **Create Resource** button.
2. Add a **GET** method to **/Questions/{id}** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **GetSingleRecord**.



1. Add a **DELETE** method to **/Questions/{id}** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **DeleteRecord**.



1. Add a **PUT** method to **/Questions/{id}** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **UpsertQuestion**.

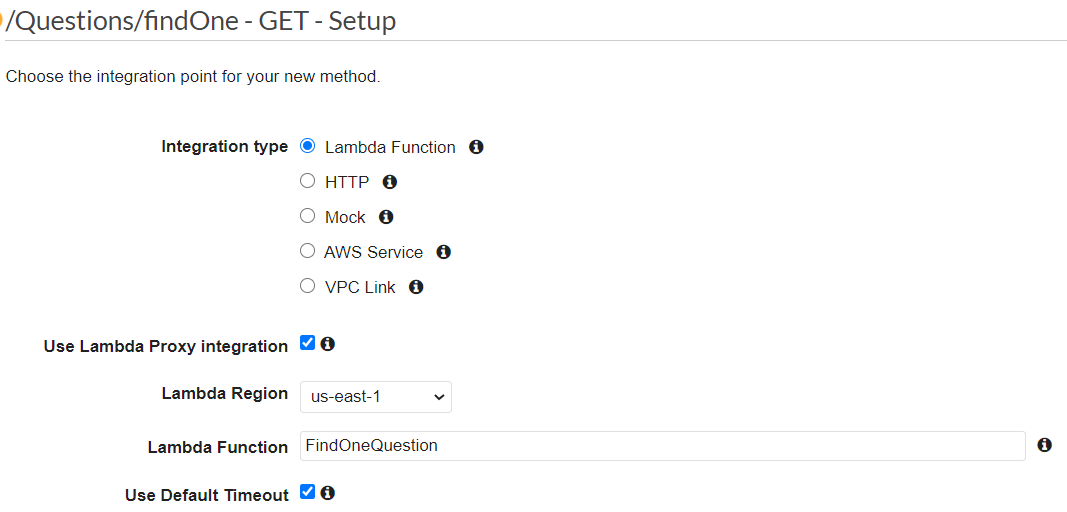


### The findOne Sub-Resource

1. Create a new resource named “findOne” under **Questions**. Pay special attention to the casing of the resource name.



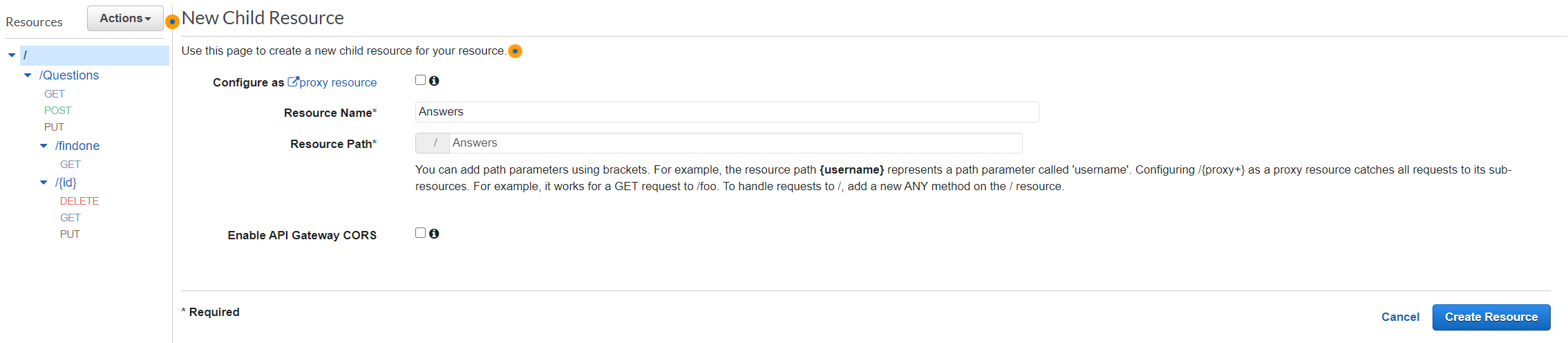
1. Add a **GET** method to **/Questions/findOne** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **FindOneQuestion**.



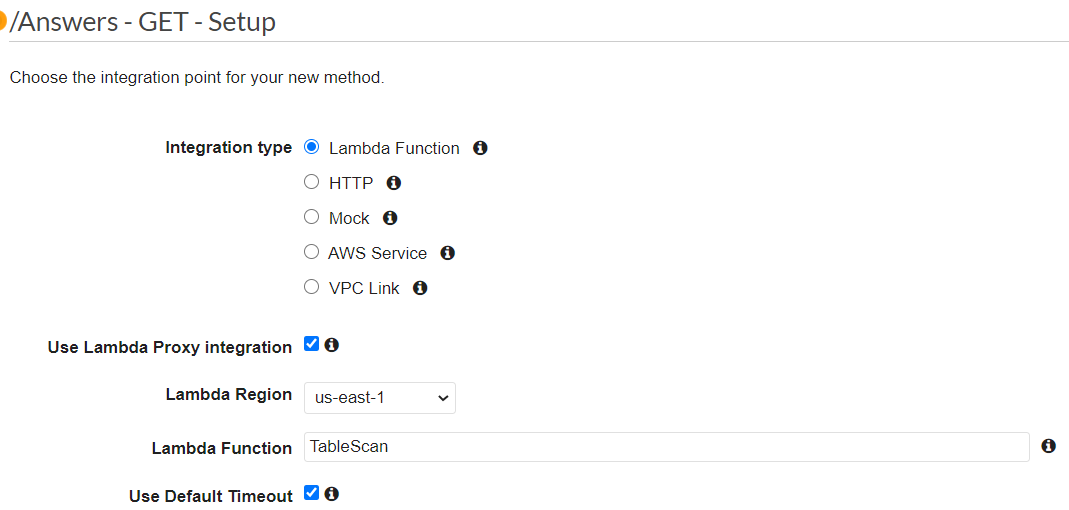
### Part Two – The Answers API

### The Answers Resource

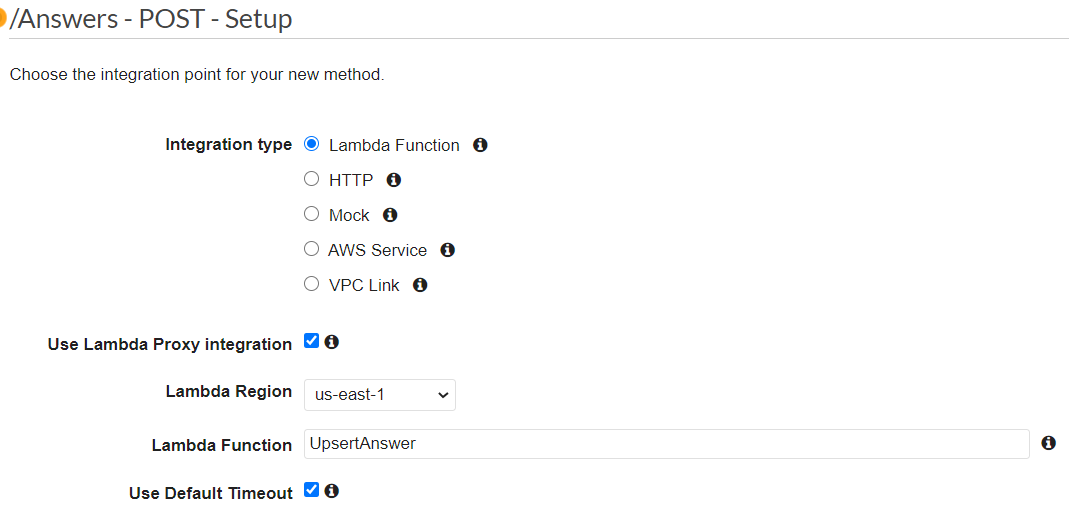
1. Create a new top-level resource (child of “/”) named **Answers**. Make sure the resource path is **/Answers**.



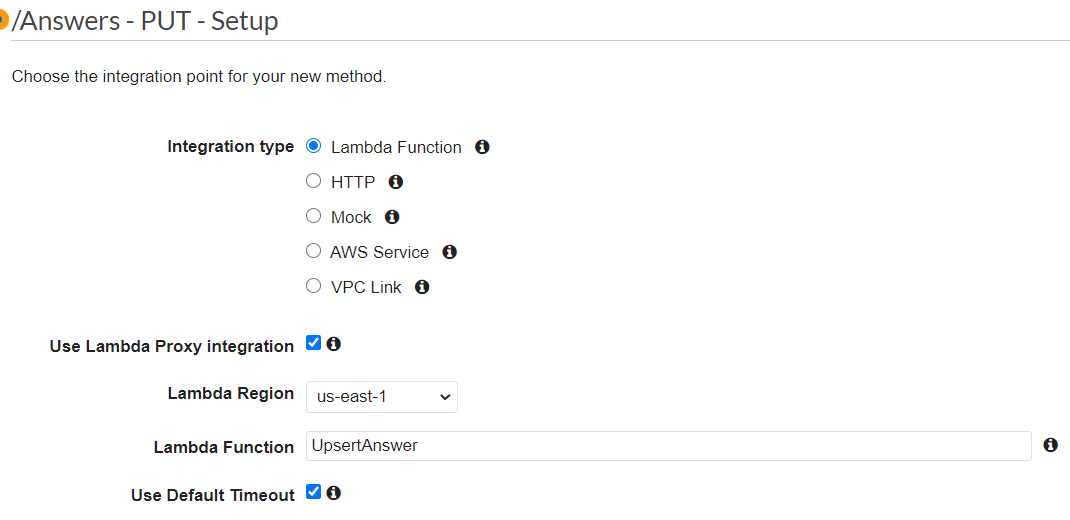
1. Add a **GET** method to **/Answers** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **TableScan**.



1. Add a **POST** method to **/Answers** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **UpsertAnswer**.

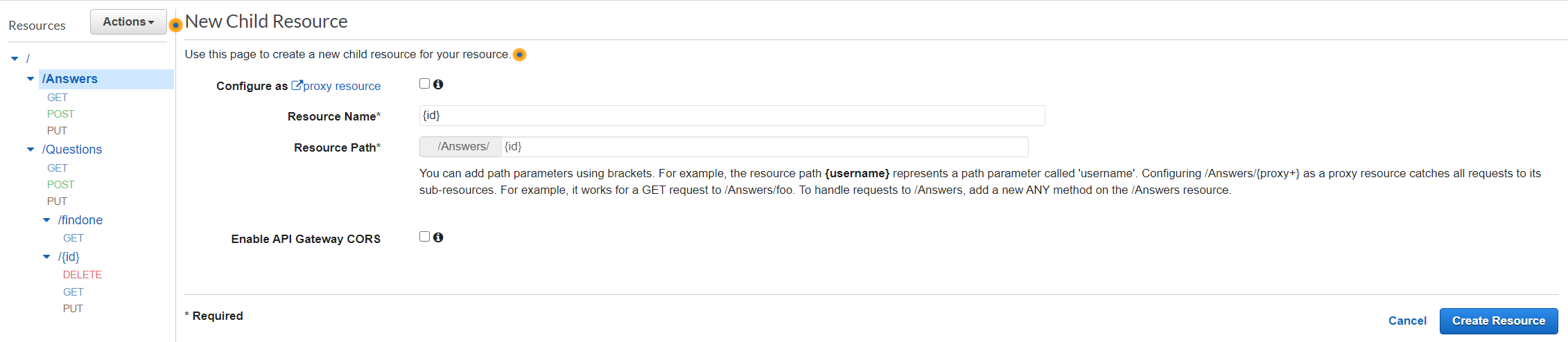


1. Add a **PUT** method to **/Answers** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **UpsertAnswer**.

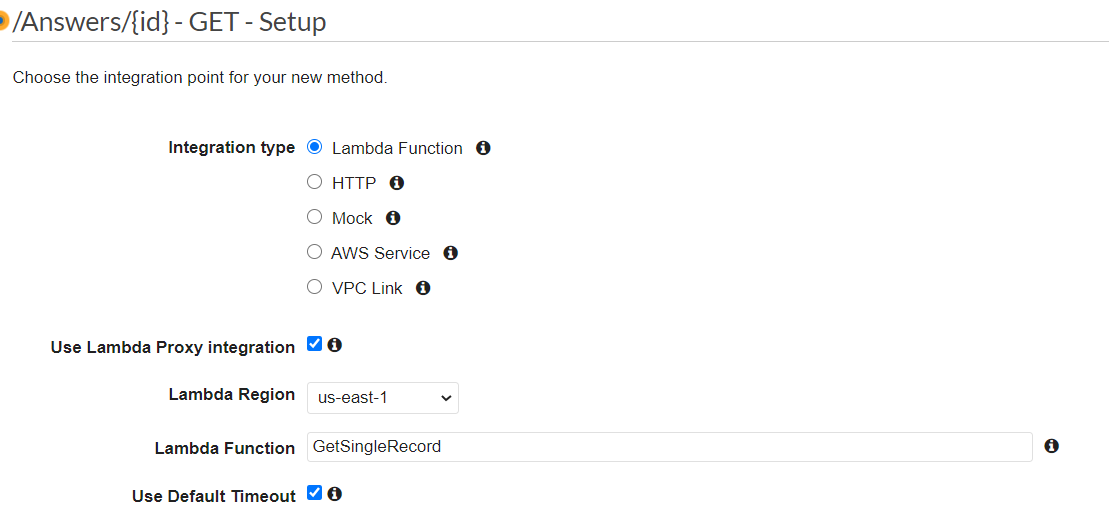


### The {id} Sub-Resource

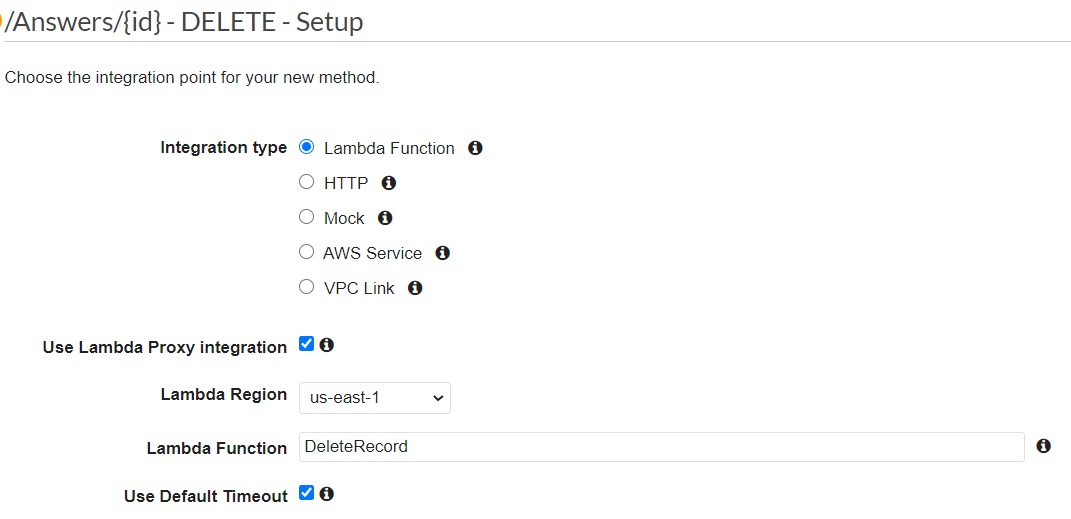
1. Create an **{id}** resource under **Answers**. Make sure the Resource Path is **/Answers/{id}**.



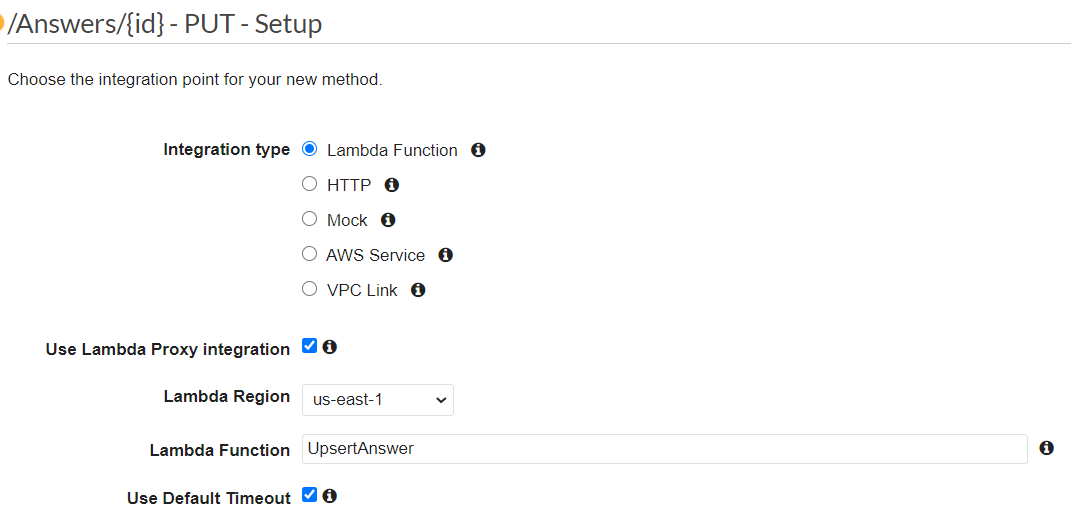
1. Add a **GET** method to **/Answers/{id}** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **GetSingleRecord**.



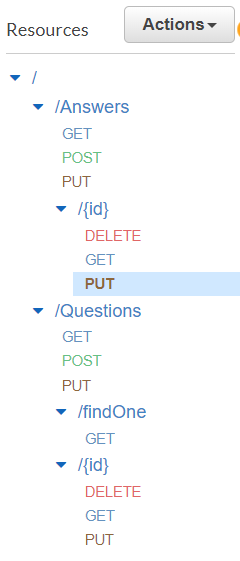
1. Add a **DELETE** method to **/Answers/{id}** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **DeleteRecord**.



1. Add a **PUT** method to **/Answers/{id}** with the following setup: a Lambda Function, using Lambda Proxy Integration, in region **us-east-1**, and with the Lambda Function **UpsertAnswer**.



1. You should now have an API Resource tree that looks like this:

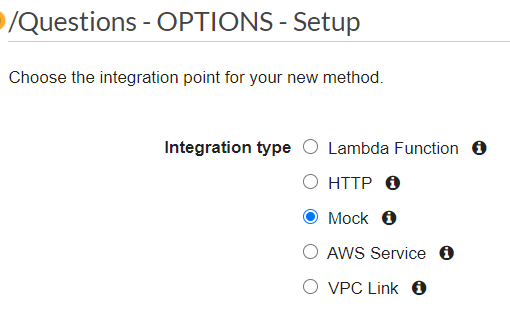


## Supporting CORS

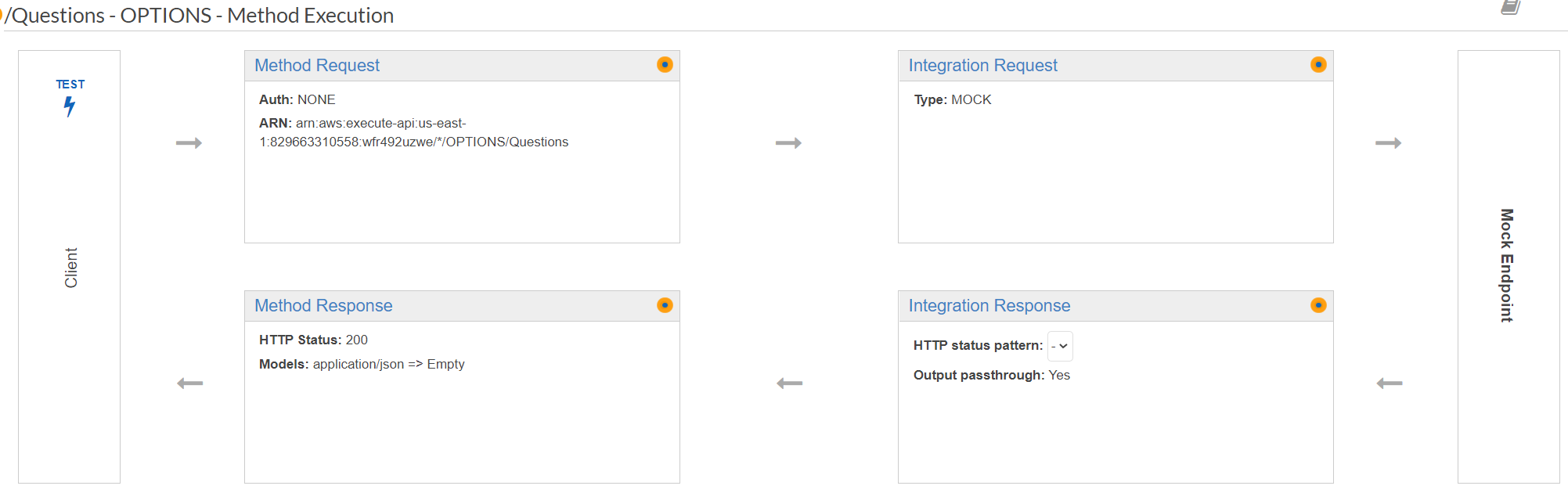
Cross-Origin Resource Sharing (CORS) is a mechanism to use HTTP headers to allow resource sharing across different domains. AWS Gateway has an action to enable CORS, but we will *not* be using it. This action is only partially successful, so we will do this manually as defined on the [CORS on AWS API Gateway](https://enable-cors.org/server_awsapigateway.html) website. The requirements are to have an **OPTIONS** method for each resource with specific headers returned, and for some specific headers to be returned on each **REST** method called.

### Adding CORS to Questions Resource

1. Select the **/Questions** resource and add an **OPTIONS** method.
2. Select **Mock** as the Integration type and click the **Save** button.



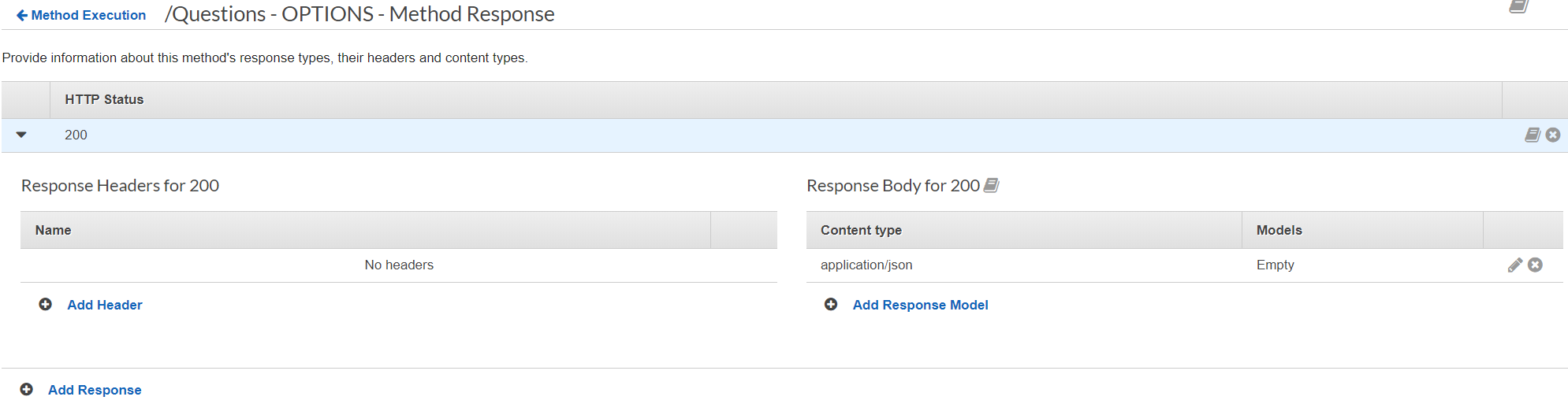
1. Click on the **Method Response** link.



1. There should be an HTTP Status of “200”, but if not, add it by clicking **Add Response**, entering “200” in the box, and clicking the checkmark on the right side.



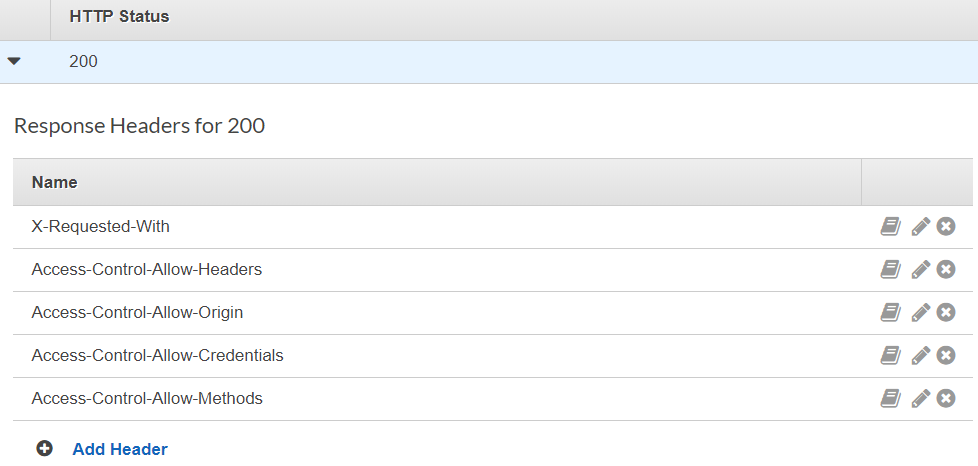
1. Click on the arrow to the left of the HTTP status to open the 200 status details.



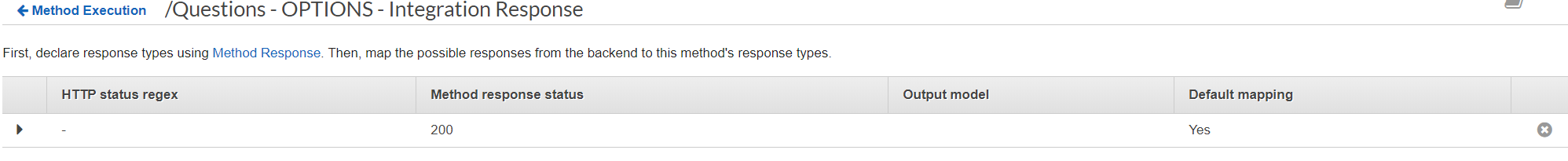
1. Now you need to add some headers. Click **Add Header**.
2. Enter “X-Requested-With” in the box and click the checkmark. Be careful to use the correct case, and not to add spaces.



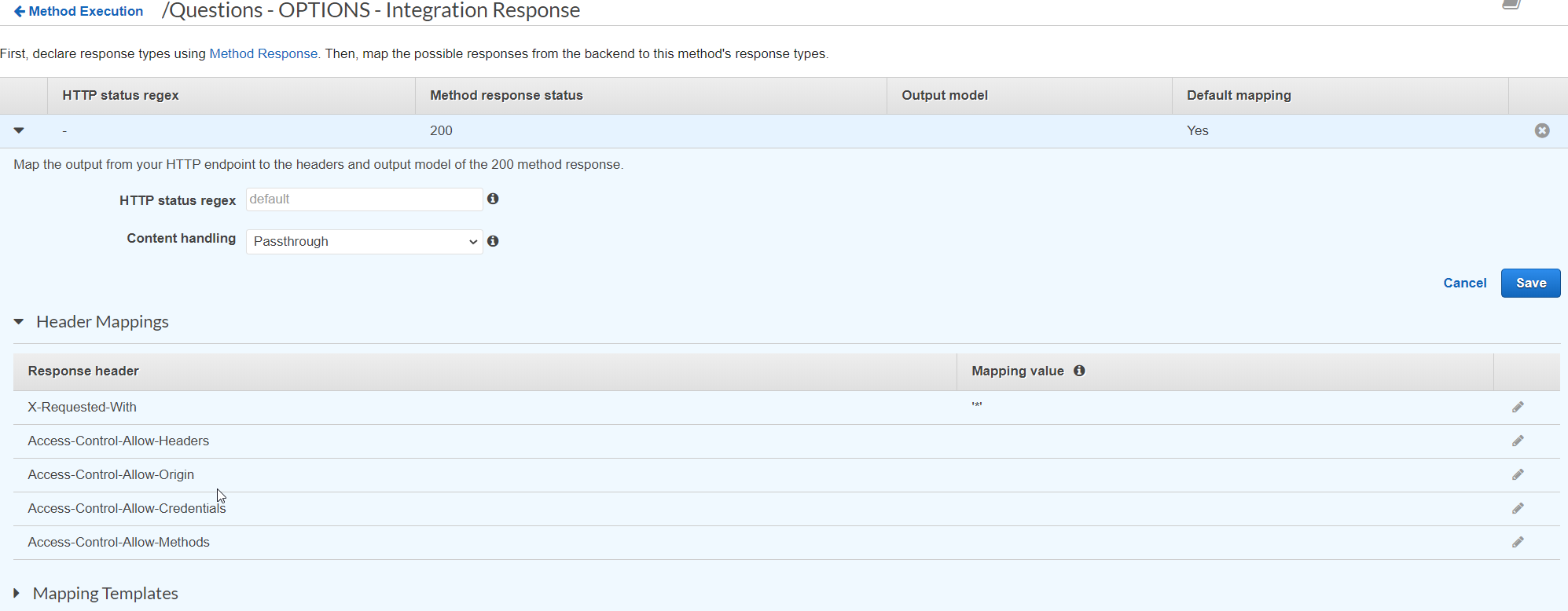
1. Repeat the process to add “Access-Control-Allow-Headers”, “Access-Control-Allow-Origin”, “Access-Control-Allow-Methods”, and “Access-Control-Allow-Credentials”. When complete, the **Response Headers** section will look like this:



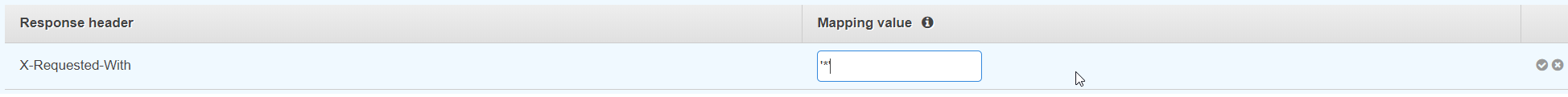
1. Click on “Method Execution” in the upper right to return to the **Method Execution** page.
2. Click the “Integration Response” link to go to the **Integration Response** page.



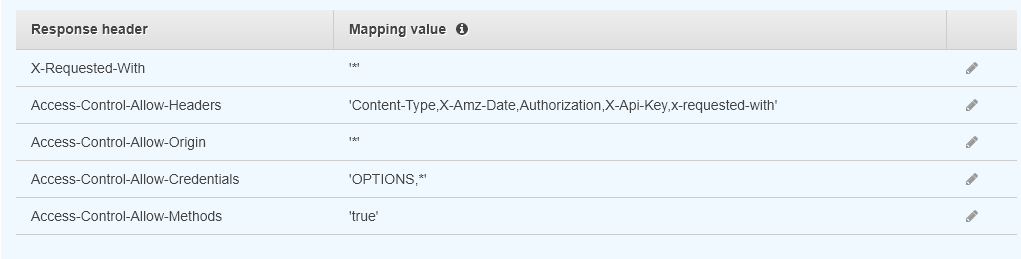
1. Open the “200” method response status by clicking the arrow to the left of the row.
2. Click on **Header Mappings** to expand that section.



1. Set the Mapping Value for each of the response headers by clicking the pencil icon on the right, entering the value, and clicking the checkmark.

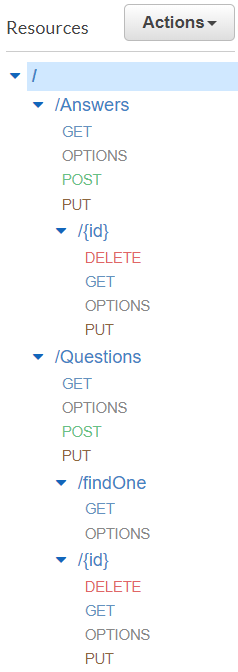


1. For **X-Requested-With**, enter ‘\*’, with the quotes.
2. For **Access-Control-Allow-Headers**, enter'Content-Type,X-Amz-Date,Authorization,X-Api-Key,x-requested-with', with the quotes.
3. For **Access-Control-Allow-Origin**, enter '\*', with the quotes.
4. For **Access-Control-Allow-Methods**, enter 'OPTIONS,\*', with the quotes.
5. For **Access-Control-Allow-Credentials**, enter ‘true’, with the quotes.
6. When done, the **Header Mappings** page should look like this:



### Adding CORS to the Remaining Resources

1. Repeat this process for each resource and sub-resource:
   * /Answers
   * /Answers/{id}
   * /Questions/findOne
   * /Questions/{id}
2. When complete, your resource tree will look like this:



### CORS Headers for the other Methods

The other methods need to return CORS headers as well. The good news is there is nothing for you to do now. Because we configured our APIs as proxies to the Lambdas, we can handle this in the Lambda code – and you did that already. Near the start of each Lambda you created in Module Five is the following code:

const responseHeaders = (headers) => {

const origin = headers.origin || headers.Origin;

return {

// HTTP headers to pass back to the client

"Content-Type": "application/json",

// the next headers support CORS

"X-Requested-With": "\*",

"Access-Control-Allow-Headers":

"Content-Type,X-Amz-Date,Authorization,X-Api-Key,x-requested-with",

"Access-Control-Allow-Origin": origin,

"Access-Control-Allow-Methods": "OPTIONS,\*",

Vary: "Origin", // for proxies

// the "has-cors" library used by the Angular application wants this set

"Access-Control-Allow-Credentials": "true",

};

};

Towards the bottom of the Lambda is the code where the response to the client is created. You had this line of code:

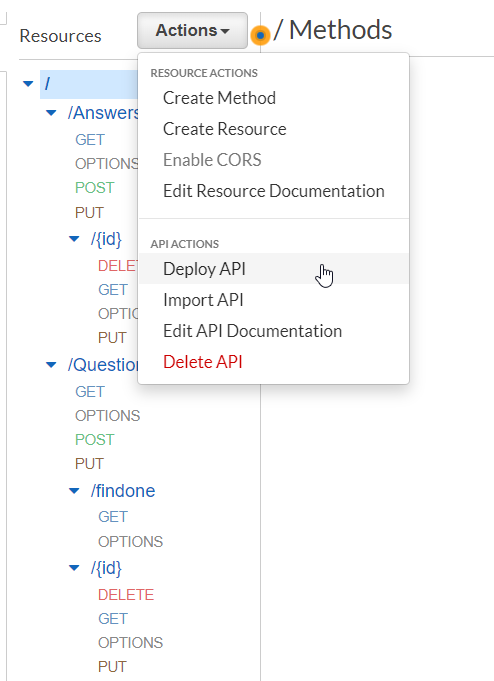
// HTTP headers to pass back to the client

headers: responseHeaders(event.headers),

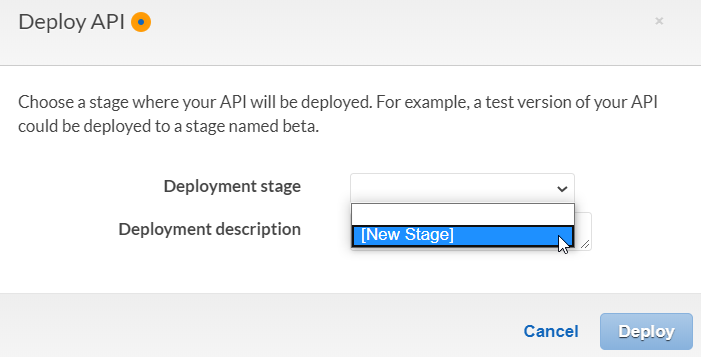
Between these two blocks of code and the APIs being configured as proxies, you are handling the CORS requirements for the other methods.

## Deployment

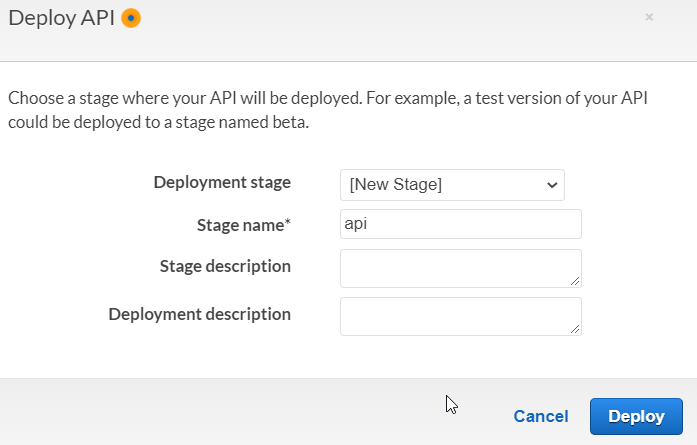
1. Deploy the API using the “api” deployment name by clicking **Deploy API** from the **Actions** menu. If you need a refresher, refer to Module Four.



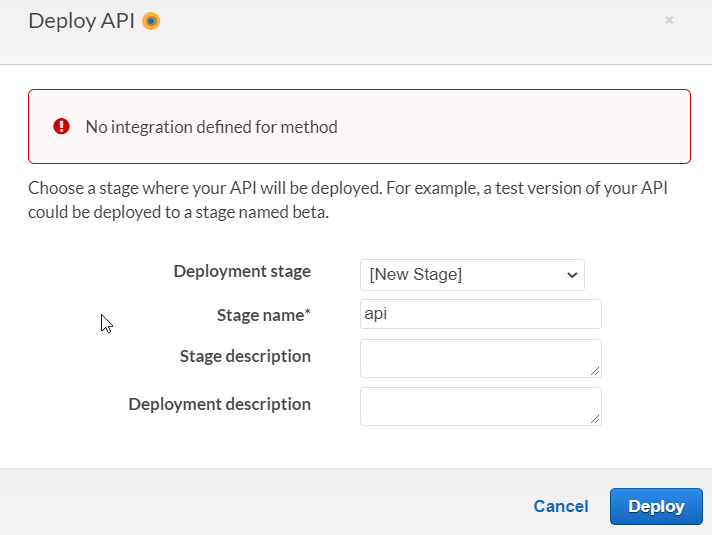
1. Create a new stage from the **Deployment Stage** drop-down menu.



1. Enter a **Stage name** of “api” and click **Deploy**.



1. A common error to see at this stage is “No integration defined for method”. If you see this error, then you forgot click the **Save** button when mapping API methods to your Lambdas. Check each API method and fix as needed. Then try again.



1. Record the **Invoke URL** displayed after deployment. It will look like this:

"api Stage Editor" screen. The Invoke URL is displayed on the right-hand side. 


## Modifying the Angular Application

1. Modify **main.ts**. Add the line **LoopBackConfig.filterOnUrl();** to the end of the file. The resulting change will look like this:

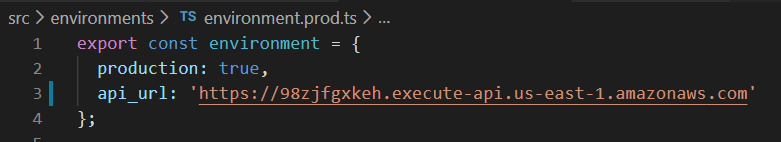
Displays the following code: 
import 'hammerjs';
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { LoopBackConfig } from '../sdk';
 
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';
 
if (environment.production) {
  enableProdMode();
}
 
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
 
LoopBackConfig.setBaseURL(environment.api_url);
 
LoopBackConfig.Url();

1. Modify **environment.prod.ts**.
   * Change the line **api\_url: 'https://q-a-example-loopback-api.herokuapp.com**:

**Displays the environment.prod.ts with the line: api_url: 'https://q-a-example-loopback-api.herokuapp.com'
**

* + To **api\_url: 'https://{deployed-url}.us-east-1.amazonaws.com’**

Remember: Your deployed URL is the one you created in the “Deployment” section of this guide. Be careful: Do not include the ‘api’ or a trailing ‘/’. The Angular application adds those.

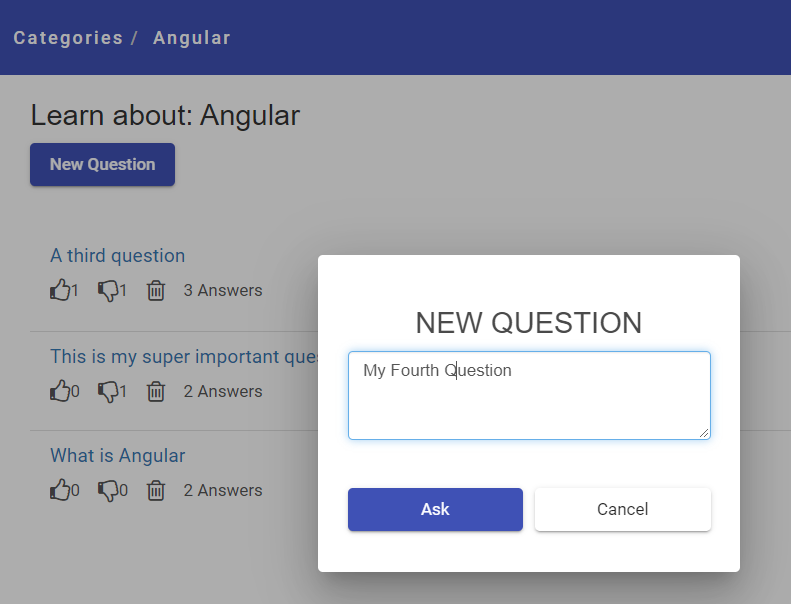
****

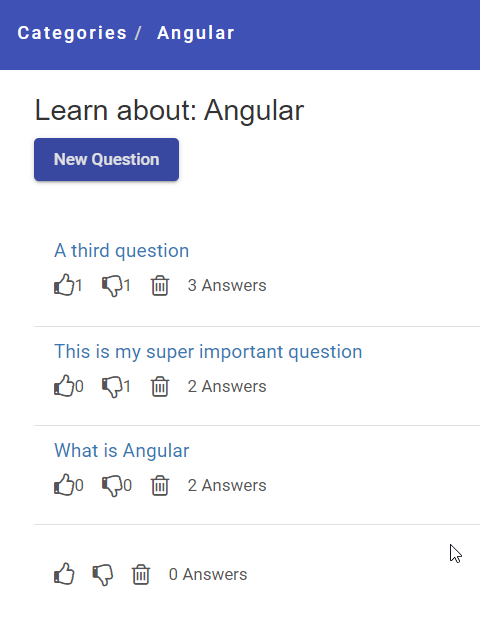
1. Build and deploy the Angular application (see the Module Three Assignment Guide).
   * Optional: You can safely delete the existing elements in your S3 bucket prior to copying new files over if you choose.

## You Are Done!

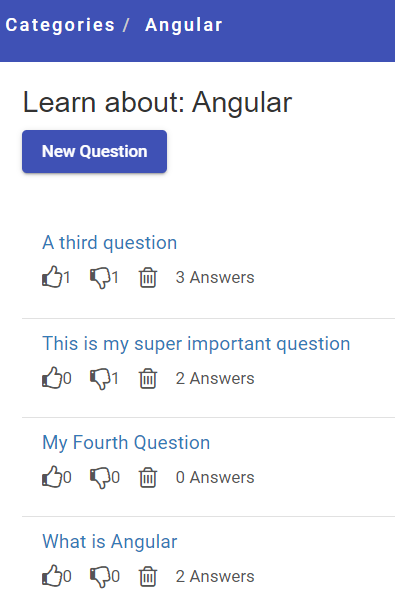
Try testing your application. Some notes to consider:

1. You shifted the application from using the Loopback SDK on the client that was tied to the Loopback REST API on the server.
2. In completing the transfer, you would change the client side to not depend on the Loopback SDK and its assumption of a Loopback REST server.
3. Why does this matter? When you add a new Question or Answer, the text of the add is not always seen right away. The call is successful, and a new entry is visible – just without the title.





1. If you click on **Categories** and back into **Angular**, the new content will show up.



However, changing that behavior may be beyond the scope of this class.

**Time to celebrate a complete project!**