The Big Picture

Learning how to use a new tool can be intimidating. Hopefully the following illustration will help remove some of that new tool anxiety.

Workflow

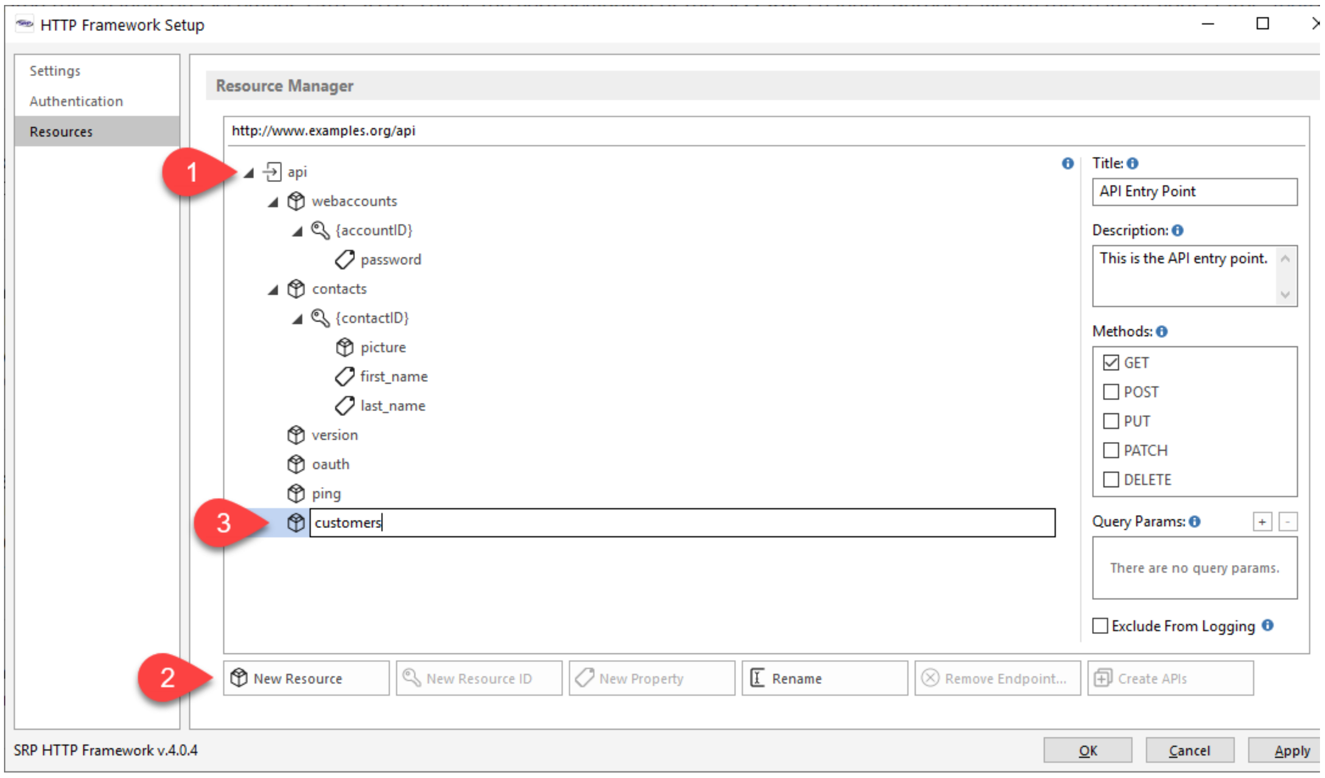
The SRP HTTP Framework basically involves two steps: **defining resources** and **creating APIs**. A *resource* definition enables the SRP HTTP Framework to determine if an incoming request is legitimate. An *API* contains the programming logic to create the digital resource that gets returned. Each of these steps can be simple or complex, depending upon the needs of the application. However, once the basic workflow steps are understood, creating rich resources and elaborate APIs becomes much easier.

Resources

Defining a Resource

Developers start by thinking about a *resource* they want to expose to the internet. A *resource* can be almost anything, but typically a *resource* will have a close relationship to a database table in the OpenInsight application. To illustrate, we'll define a *resource* that relates to a CUSTOMERS database table.

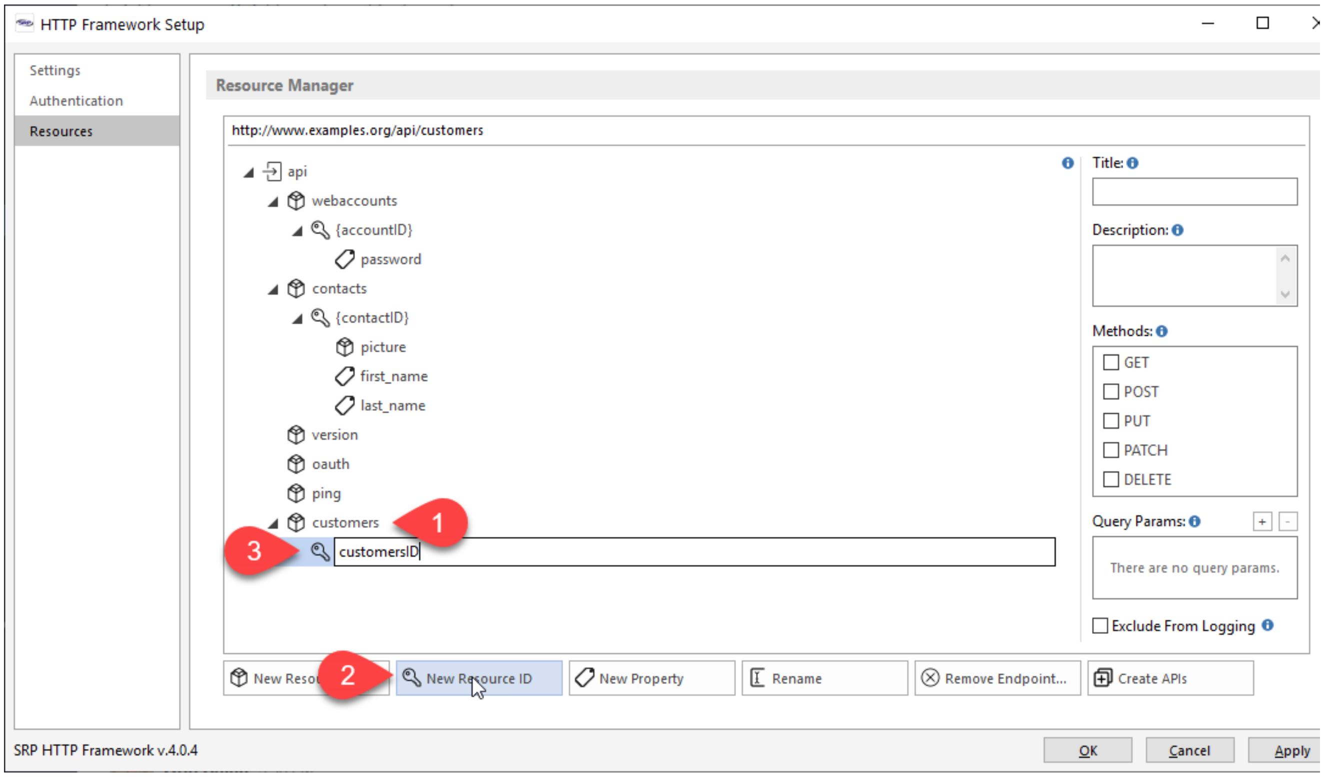
We'll use the **Resource Manager** to define a *resource*. Most resources will appear directly underneath the API endpoint (i.e., they are *primary* resources). To do this we *(1)* select the *api* resource node from the *Resource Manager* tree view, *(2)* click on the *New Resource* button, and *(3)* enter the name of the new *resource*:



Defining a Resource ID

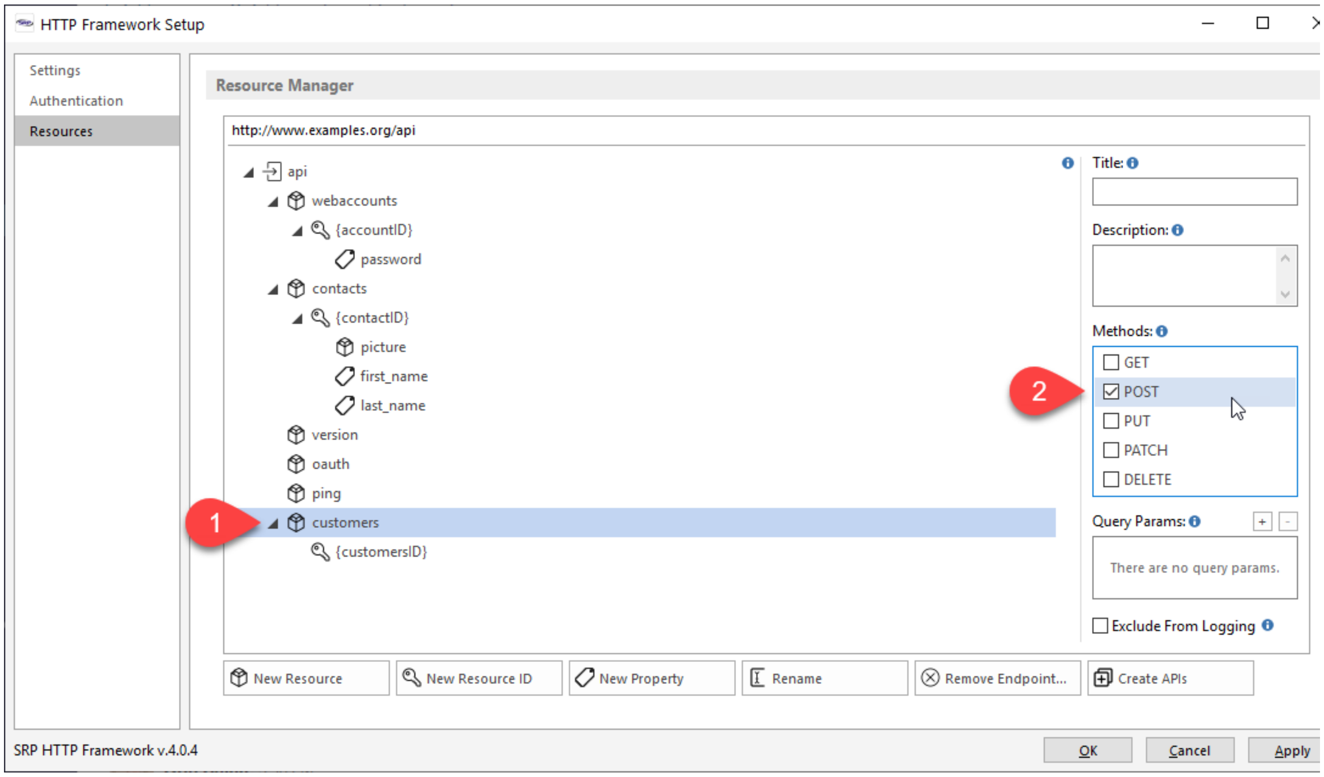
Is the resource we just defined *singular* or is it a *collection* (aka *parent*) of other *child* resources? In our case, the *customers* resource is a *collection* because it represents multiple individual customers, each with their own unique identifier. Therefore, we need to define a *resource ID* so any specific customer resource can be identified.

We start by *(1)* selecting the newly created *customers* resource node from the *Resource Manager* tree view, *(2)* click on the *New Resource ID* button, and *(3)* enter the name of the *resource ID*:

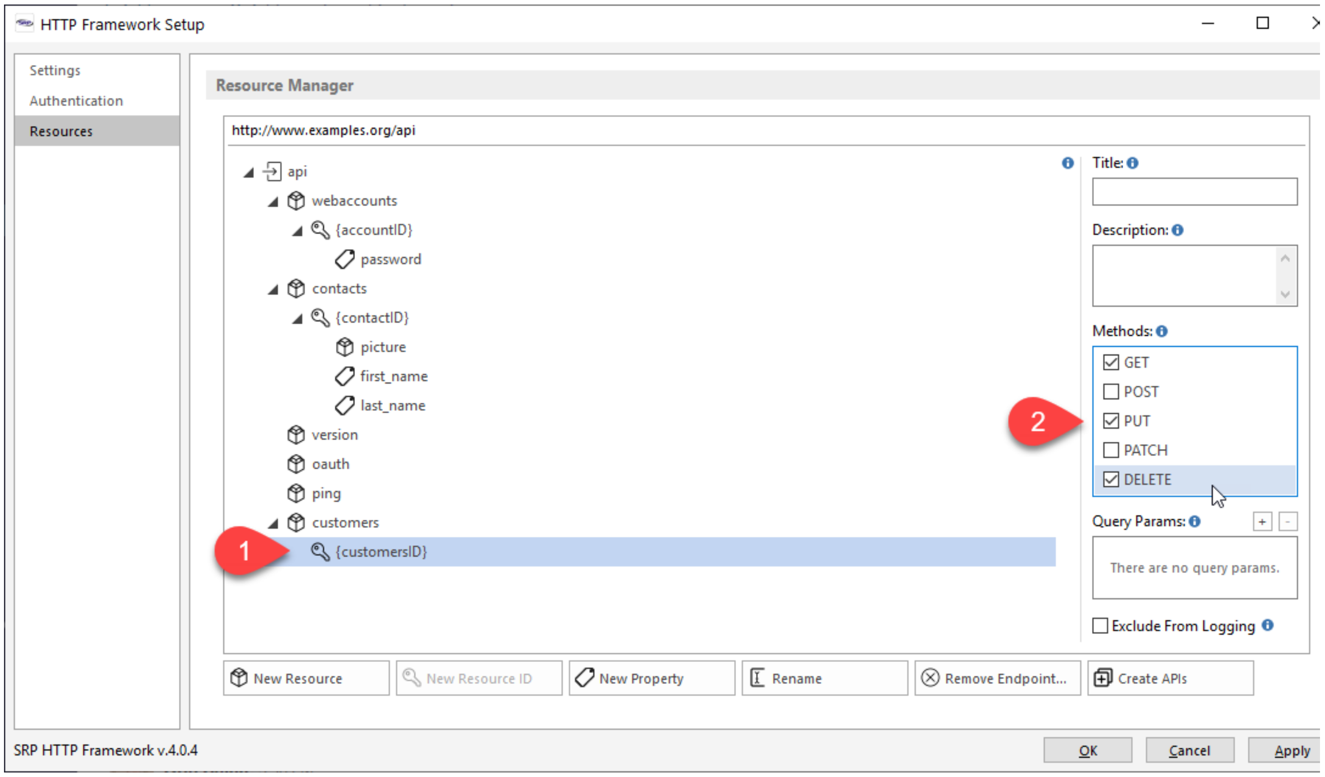


Enabling our Methods

Once a resource endpoint has been defined, we need to define how clients will be able to interact with it by enabling one or more *methods*. For most database driven resources, CRUD functionality is achieved through the *POST* (create), *GET* (read), *PUT* (update), and *DELETE* (delete) *methods*. For our purposes, we'll allow new customers to be created by enabling the *POST* method for the *customers* resource endpoint. To do this we just *(1)* select the *cust omers* resource node from the *Resource Manager* tree view and *(2)* click on the *POST* checkbox under the **Methods** block:



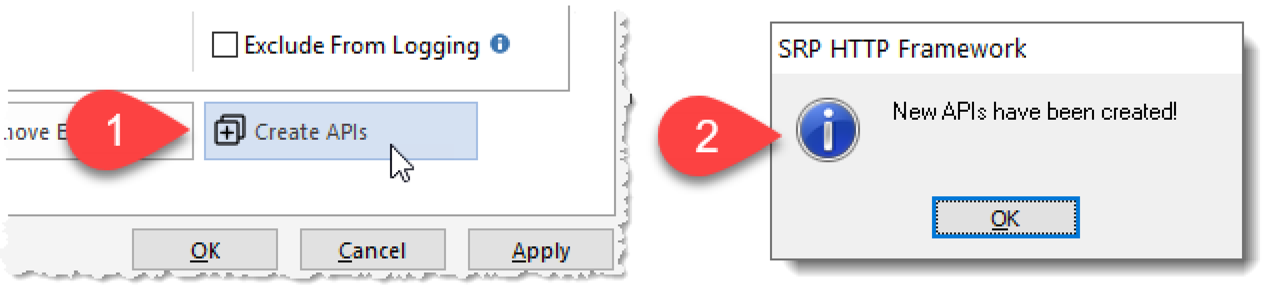
We'll also allow specific *customerID* resource endpoints to be read, updated, and deleted. To do this we *(1)* select the *customerID* resource node from the *Resource Manager* tree view and (2) click on the *GET, PUT, and DELETE* checkboxes under the **Methods** block:



APIs

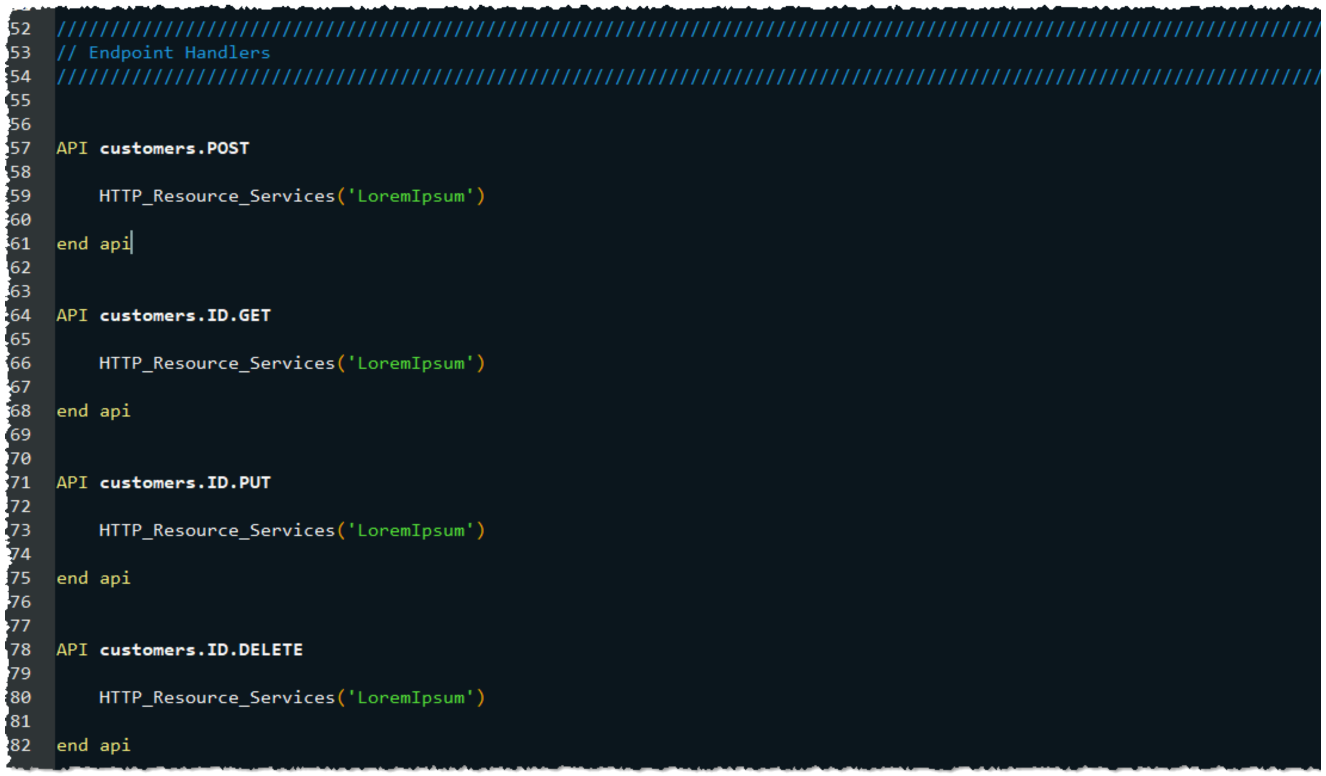
Creating our API Commuter Module

When we are happy with our new resource endpoints, we create the *API commuter module* simply by *(1)* clicking on the *Create APIs* button. This should *(2)* produce a dialog message confirming that new APIs have been created:



Editing our API Commuter Module

All *API commuter modules* are named after the parent resource. Resource IDs are not considered an independent resource, even though it has its own endpoint, so their APIs will always be included within the *API commuter module* of their parent resource. In our case, the parent resource is *customers* so our *API commuter module* will be called *CUSTOMERS_API*. Let's open it using the SRP Editor:


```
////////////////////////////////////////////////////////////////////////////////////////////////////
// Endpoint Handlers
////////////////////////////////////////////////////////////////////////////////////////////////////


API customers.POST

    HTTP_Resource_Services('LoremIpsum')

end api


API customers.ID.GET

    HTTP_Resource_Services('LoremIpsum')

end api


API customers.ID.PUT

    HTTP_Resource_Services('LoremIpsum')

end api


API customers.ID.DELETE

    HTTP_Resource_Services('LoremIpsum')

end api
```

The above is just a snippet of the *CUSTOMERS_API* commuter module, but it showcases all of our customer resource APIs:

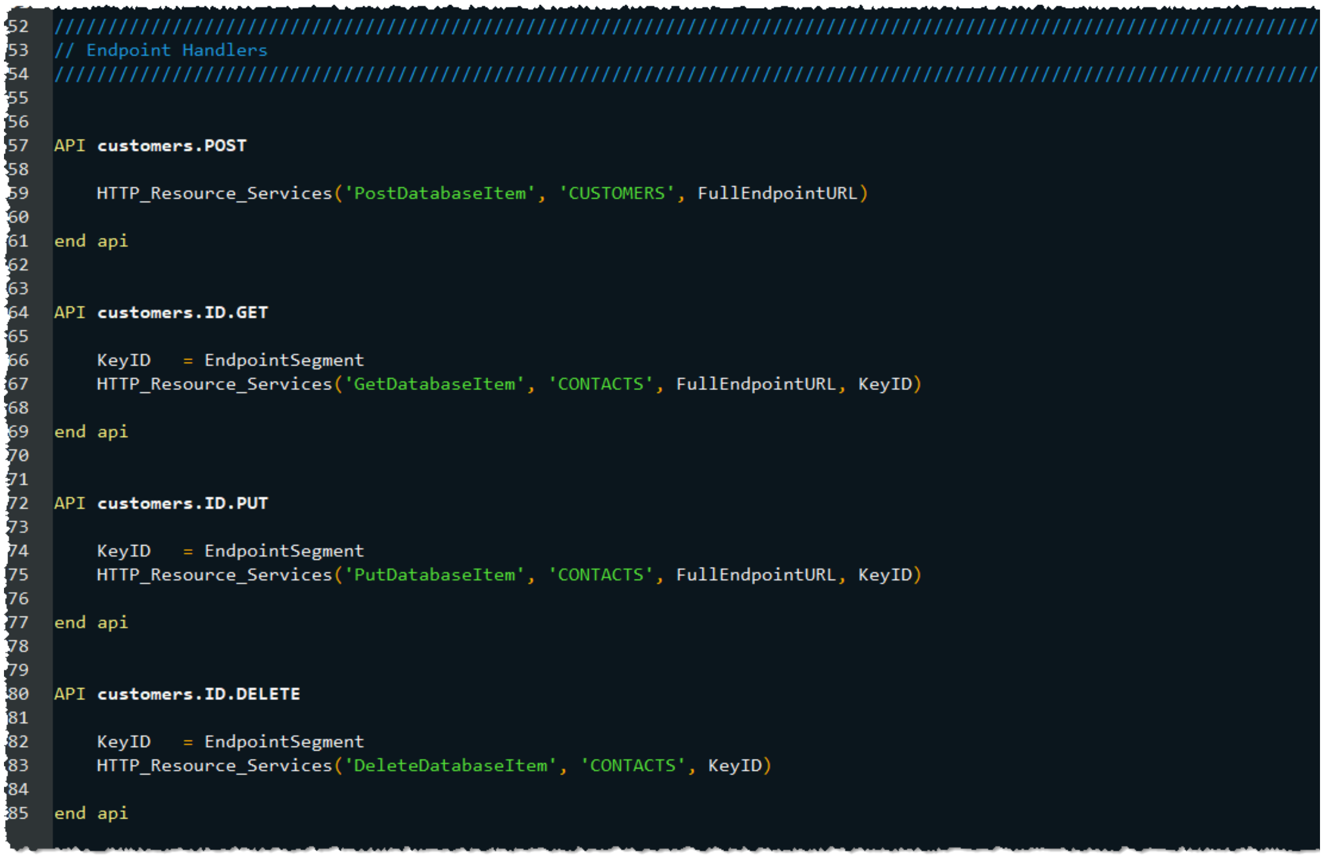
API Signature	Purpose
`API customers.POST`	Creates a new customer.
`API customers.ID.GET`	Reads a specified customer .
`API customers.ID.PUT`	Updates a specified customer.
`API customers.ID.DELETE`	Deletes a specified customer.

In each of our APIs there is a call into the *LoremIpsum* service (a member of the HTTP_Resource_Services module). The purpose of this special service is to enable the new API to produce sample content so the API can be tested immediately (see our *How do I create an API?* article for an example of this).

Making APIs Functional

We'll assume that our APIs are responding properly to requests so we are now ready to make our *APIs function* the way we want. The way this gets done can vary greatly based on the nature of the resource, the purpose of the API, and the overall design intent of the application. In simple cases where the resource is related to a database table, we can use some high-level services to make this quick and easy. Since our customers resource is related to the CUSTOMERS database table, we'll update our API commuter module as follows:

Obviously there is a lot of automation going on within these services. Depending upon your needs, your code might look very different. We encourage you to start with our *How do I create a resource?* article to get a feel for the different ways this task can be approached.



```
////////////////////////////////////////////////////////////////////////////////////////////////////////
// Endpoint Handlers
////////////////////////////////////////////////////////////////////////////////////////////////////////


API customers.POST

    HTTP_Resource_Services('PostDatabaseItem', 'CUSTOMERS', FullEndpointURL)

end api


API customers.ID.GET

    KeyID   = EndpointSegment
    HTTP_Resource_Services('GetDatabaseItem', 'CONTACTS', FullEndpointURL, KeyID)

end api


API customers.ID.PUT

    KeyID    = EndpointSegment
    HTTP_Resource_Services('PutDatabaseItem', 'CONTACTS', FullEndpointURL, KeyID)

end api


API customers.ID.DELETE

    KeyID    = EndpointSegment
    HTTP_Resource_Services('DeleteDatabaseItem', 'CONTACTS', KeyID)

end api
```

In Summary

Hopefully this illustration encourages you to use the SRP HTTP Framework with a sense of confidence. Quite often the developer just repeats the above steps as new resources are added to the application or when existing resources are updated (e.g., adding a new method an endpoint). There are other important elements that this article did not explore, such as authentication, authorization, unique resource media types, hypermedia, etc., but these will all come in good time and will be easier to implement once you master the basics.