**Definition**

An **API** is an application programming interface. It is a set of rules that allow programs to talk to each other. The developer creates the API on the server and allows the client to talk to it.

**REST** determines how the API looks like. It stands for “Representational State Transfer”. It is a set of rules that developers follow when they create their API. One of these rules states that you should be able to get a piece of data (called a resource) when you link to a specific URL.

**Anatomy of an API**

Let us first understand the anatomy of a URL(Uniform Resource Locator).

A URL comprises of the following parts:

**Endpoint**: Address of the server

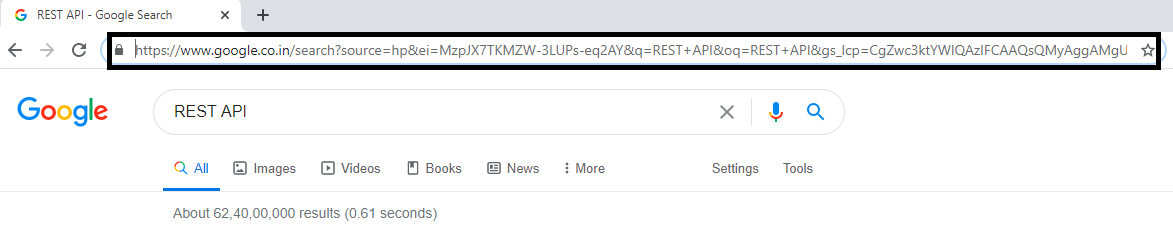
**Resource/Route**: Identifier of the service being consumed

**Parameters**: Additional variables/filters(name value pairs)

For example, lets run a basic search on google and try to identify each of these parts:

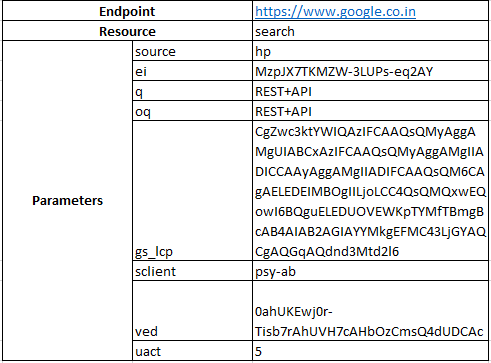
|  |
| --- |
|  |

On pressing enter, browser open a page with following URL:



<https://www.google.co.in/search?source=hp&ei=MzpJX7TKMZW-3LUPs-eq2AY&q=REST+API&oq=REST+API&gs_lcp=CgZwc3ktYWIQAzIFCAAQsQMyAggAMgUIABCxAzIFCAAQsQMyAggAMgIIADICCAAyAggAMgIIADIFCAAQsQM6CAgAELEDEIMBOgIILjoLCC4QsQMQxwEQowI6BQguELEDUOVEWKpTYMfTBmgBcAB4AIAB2AGIAYYMkgEFMC43LjGYAQCgAQGqAQdnd3Mtd2l6&sclient=psy-ab&ved=0ahUKEwj0r-Tisb7rAhUVH7cAHbOzCmsQ4dUDCAc&uact=5>

Let us now dissect the URL:



Now that we understand a URL properly, let us understand the anatomy of a REST API.

Basically, a REST API is a URL packaged with following additional information:

**Method Type** : GET/POST/PUT/DELETE/PATCH

**Authorization** : Logging credentials

**Headers** : It contains additional information variables with value(NVP) to be passed while requesting a service.

**Body(Data)** : This can be used to pass additional variables as form fields(NVP) as well as complex information such as tabular-data(array), nested structures, deep structures etc.

**Creating a REST API in SAP**

For creating REST API in SAP with in built X-CSRF token authentication we need to create one HANDLER class for endpoint and one or more RESOURCE classes for managing the child routes.

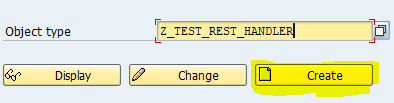
The job of HANDLER class is to receive the request, authenticate X-CSRF token and then call the RESOURCE class mapped for the respective route.

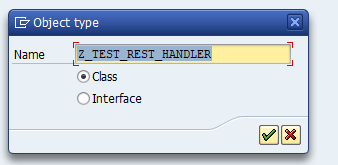
The job of a RESOURCE class attached to a route is to process the request methods (GET/POST/PUT/DELETE/HEAD/OPTIONS) and return response to the client.

Although, for multiple routes belonging to same endpoint, its possible to use the same RESOURCE class if handled properly, its rather recommended to create independent RESOURCE classes for each route. This ensures that these classes are independently deployable, testable, maintainable and could be owned separately by developers. With this approach, the API endorses the core principle of **Microservice Architecture**.

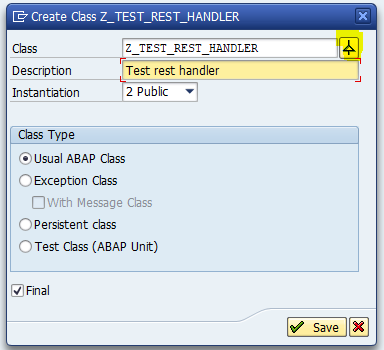
**Steps** :

1. Create Handler class:
2. Go to t-code SE24 and create a class

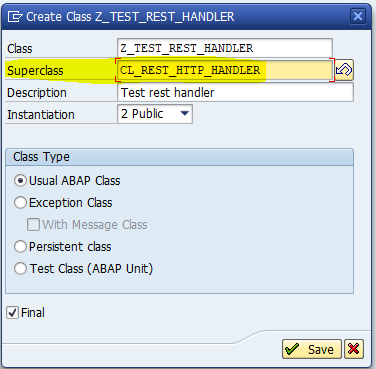




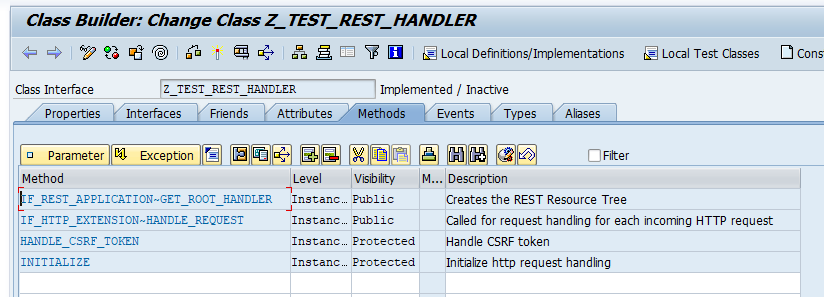
1. On the next pop-up screen provide description and click on “Create Inheritance” icon:



1. Enter superclass name: CL\_REST\_HTTP\_HANDLER and save

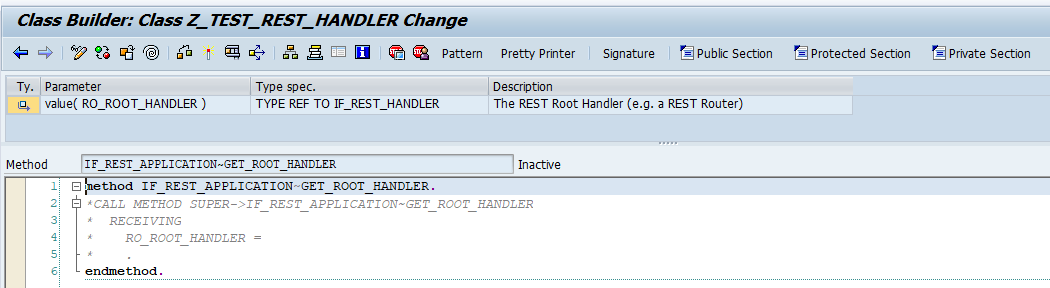


We should be able to see following 4 methods inherited from parent class:



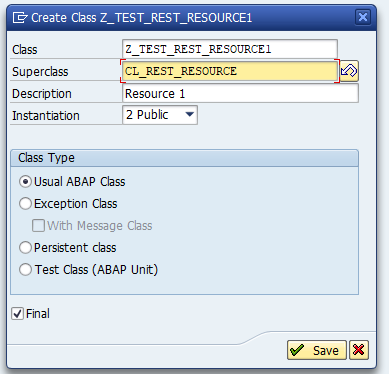
Please note that GET\_ROOT\_HANDLER is an abstract method and hence it is mandatory to redefine the method.

Let us leave the body of the method blank and activate the class for now. We will add our implementation soon.

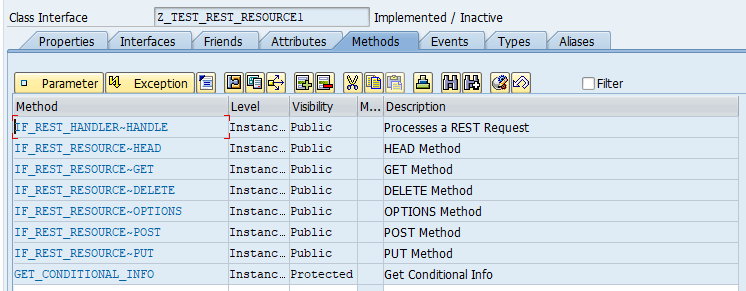


1. Create RESOURCE class:

For creating a RESOURCE class, we need to inherit class “CL\_REST\_RESOURCE”:

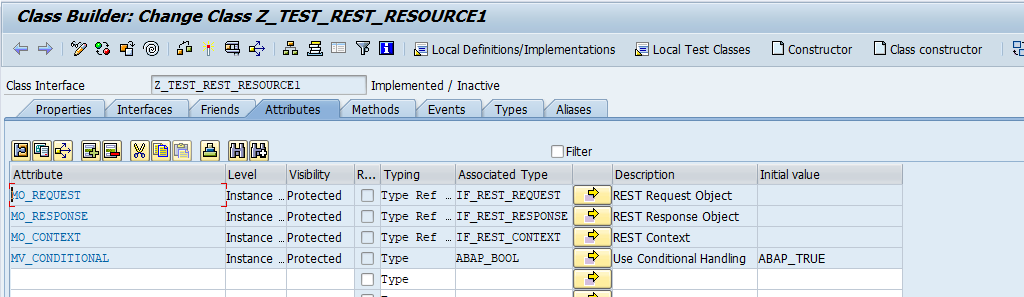


On save, we should be able to see following methods:

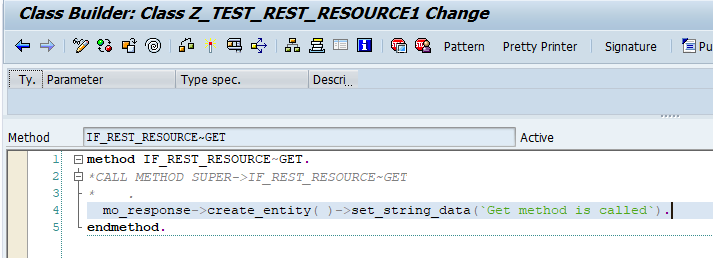


Unlike CL\_HTTP\_REST\_HANDLER, all these methods are implemented in parent class where each of these methods raise an exception “Not allowed” by default. This ensures that only those methods are valid which are implemented by us(developer).

The request data is available in the global attribute MO\_REQUEST and response if required to be sent can be populated using attribute MO\_RESPONSE.

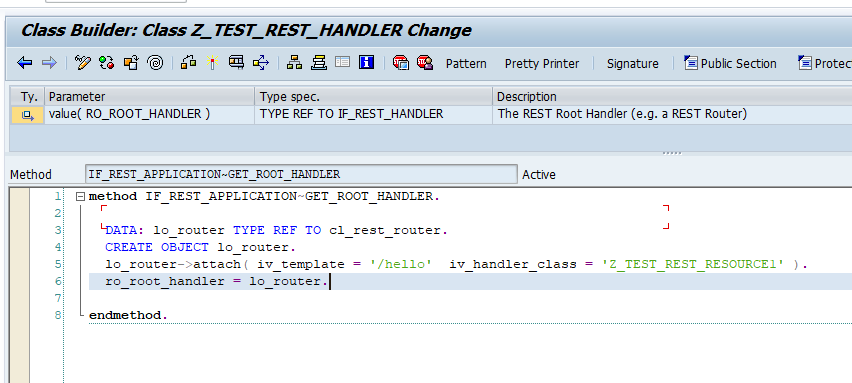


Let us re-define GET method and POST method with a simple message as response body:

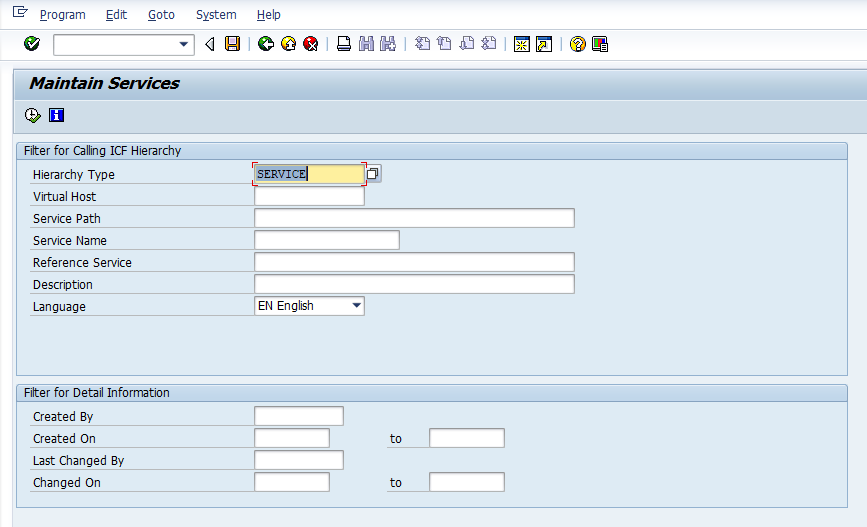


1. Attach the RESOURCE class to a route in HANDLER class:

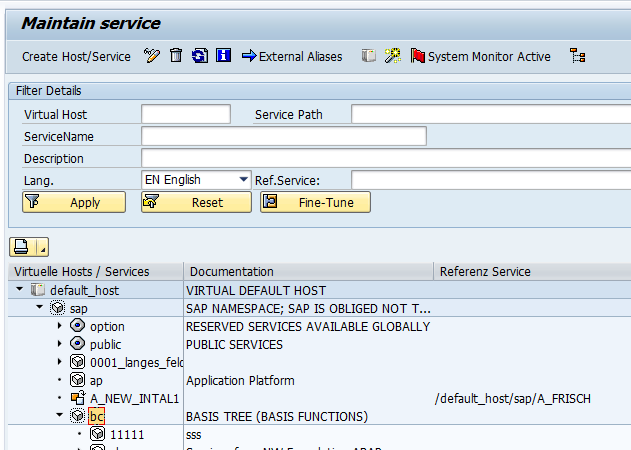
Let us go back to implementation of GET\_ROOT\_HANDLER and attach route to resource.



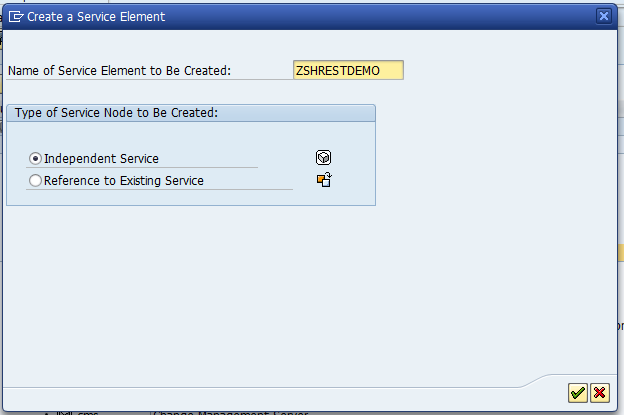
1. Create ICF service for our API.
2. Run t-code SICF



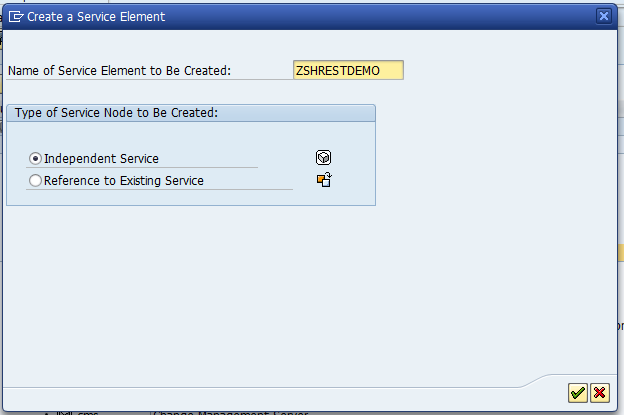
1. Execute with Hierarchy type as “SERVICE”
2. Navigate to default\_host/sap/bc



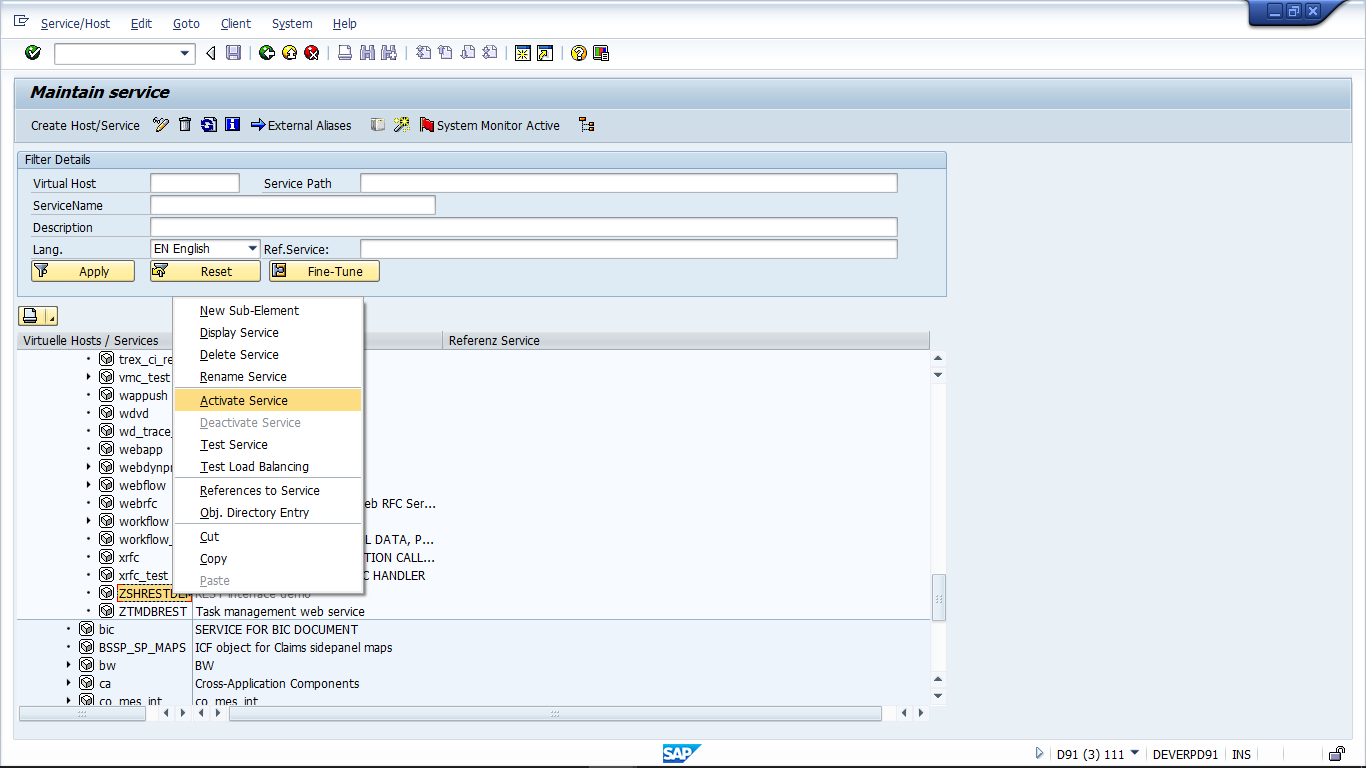
1. Right click on “bc” and click on “New sub-element”
2. Click ok on information message window
3. Enter name of Service and select radio button “Independent Service”



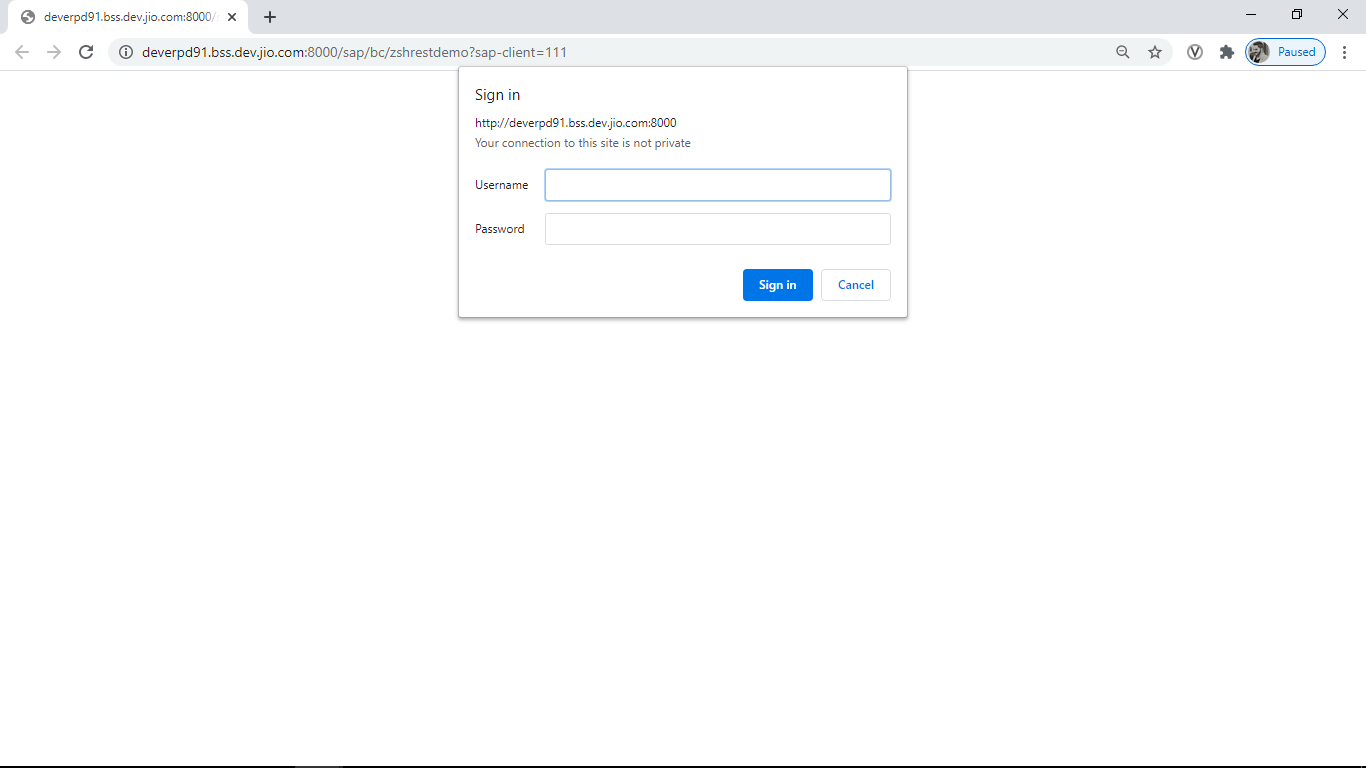
1. Right click on “bc” and click on “New sub-element”
2. Click ok on information message window
3. Enter name of Service and select radio button “Independent Service”

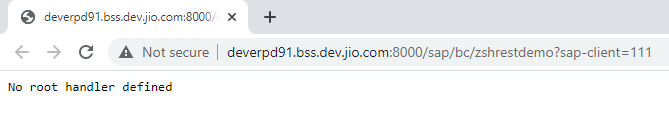


1. Enter implemented HANDLER class name in Handlers List
2. Click on back and activate created sub-element



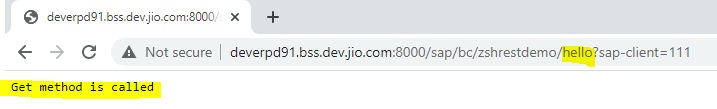
1. Right click on sub-element and click on Test service. Enter the credentials in pop-up window:





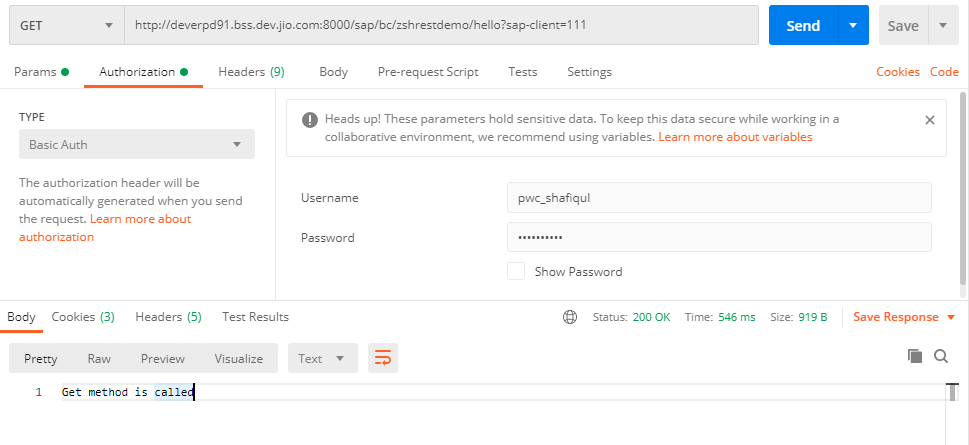
The error received is due to no mapped RESOURCE class for route “ “.

However, we have mapped route “/hello” to a resource class so let us try that route.



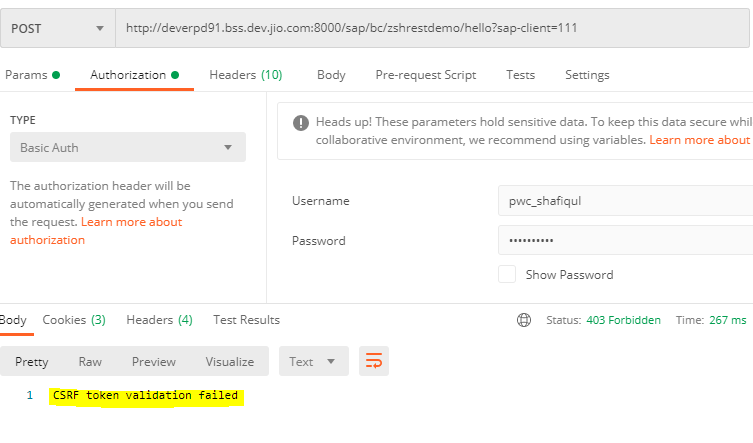
As we can see, the GET method is called properly.

Now let us try to call the GET method from POSTMAN:



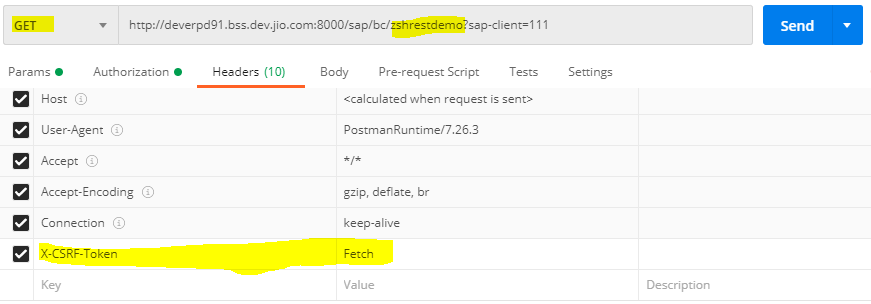
The method is called successfully.

Now let us try to call the POST method from POSTMAN.

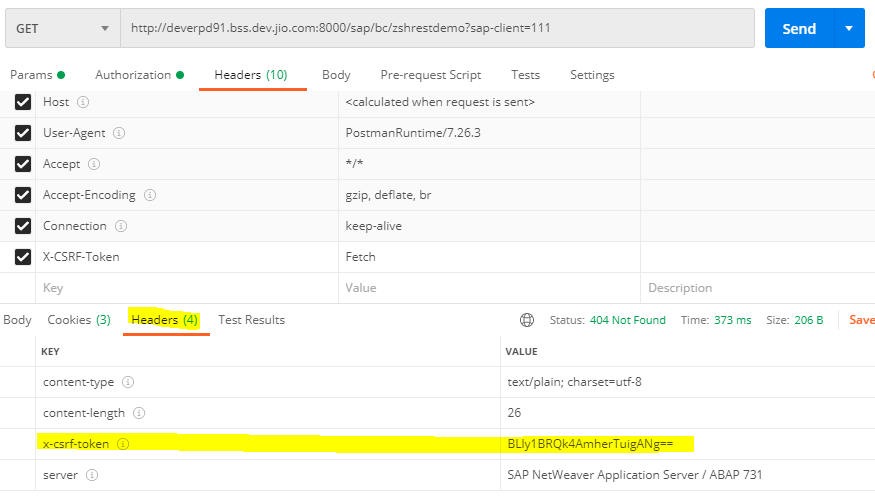


As we can see, POST call failed with error “CSRF token validation failed”.

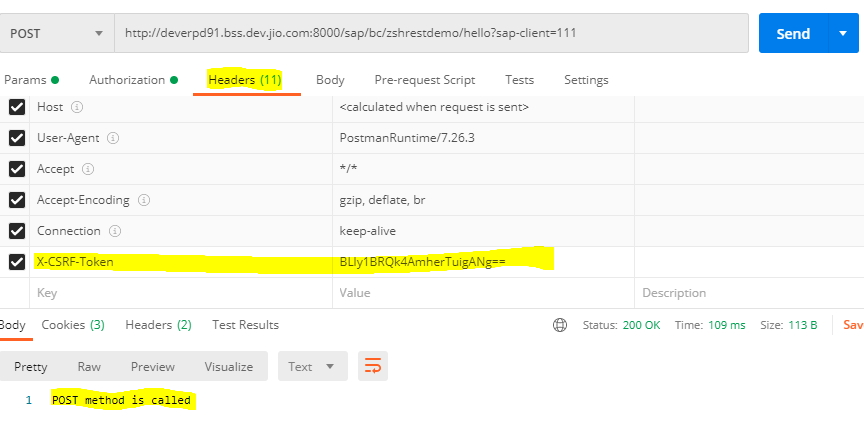
To call POST/PUT/DELETE method, we need to first fetch a valid X-CSRF token from this endpoint or any valid endpoint of the server using GET method and header field “{X-CSRF-Token:Fetch}”.



In the response header tab, we should be able to find generated token:



Now, in the POST call we need to pass this token as HEADER field:



As we can see, POST method is successfully called.