REST API Service creation in SAP

# 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).

## Anatomy of a URL

A URL comprises of the following parts:

Endpoint : Address of the server and PORT.

Resource/Route : Identifier of the service being consumed.

In API context a resource is generally any DB table on which the CRUD operations are performed. The route could be named same as the RESOURCE. However, API developers have the liberty to name the route mapped to a resource, they find most appropriate. For example, if the DB table name is EMPDETAILS, developer may chose the name of the route as “/empdetails” or let the name “/empdet” or “/edetails” or “employee-det” or any similar name that makes sense and uniquely identifies a service being performed.

Parameters(PARAMS) : Additional variables/filters(name value pairs(NVP)).

Can be used to filter the result set. For example, we want to read data from EMPDETAILS for a specific state and gender. In such case, state and gender could be passed at the suffix of the URL using ‘?’ as the start point and ‘&’ as a separator.

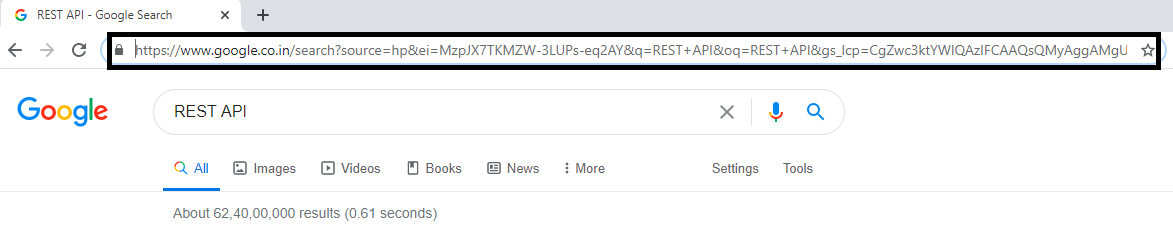
../empdetails?state=MH&gender=F

To have more clarity let us understand by a day-to-day example.

Lets run a basic search on google and try to identify each of these parts for the returned URL.

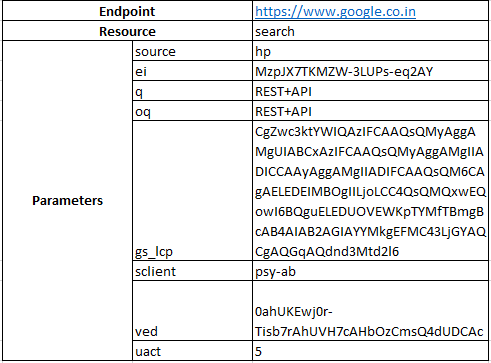
|  |
| --- |
|  |

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:



## Additional datasets of a REST API

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

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

Method Type : GET/POST/PUT/DELETE/PATCH/HEAD/ OPTONS

Authorization : Authentication type/ Logging credentials

Headers : It contains additional information variables with value(NVP) to be passed while requesting a service. Similar to params but not part of the header.

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 the following three elements.

## ICF Service

Internet Communication Framework (**ICF**) is a programming interface (API), based on interfaces and classes, used by **ABAP** programs to communicate with the Internet. **ICF** supports the Internet protocols HTTP, HTTPS, and SMTP. All communication between **ABAP** and the Internet based on these protocols takes place using **ICF**.

An ICF service configuration specifies the path of the endpoint on server, logon procedures such as Password less/Basic/Certificate based, the handler class which will further handle the request.

## HANDLER Class

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.

## RESOURCE Class

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 resource 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**.

## Understanding CSRF/XSRF

### What is CSRF/XSRF?

Cross-Site Request Forgery (CSRF), often referred as XSRF is an attack that forces an end user to execute unwanted actions on a web application in which they’re currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker’s choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

## What is X-CSRF-Token validation?

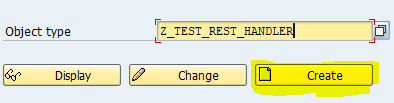
To prevent CSRF attacks, X-CSRF token validation could be implemented. In this approach, any HTTP method which is not ‘GET’ such as POST/PUT/DELETE, can only be serviced when a valid X-CSRF token is provided along with the requests. X-CSRF token can be obtained by accessing any valid endpoint on the server(Explained in the subsequent topics).

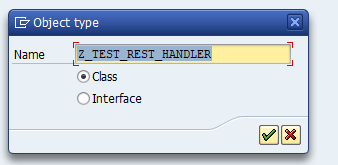
This also suggests that GET method should not be used for any service that has business logic implemented, irrespective of the DB operation type(CRUD) performed.

## Steps to create REST API

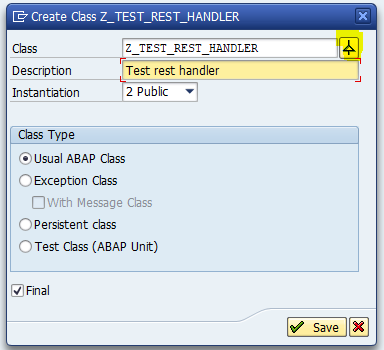
### Create Handler class:

1. Go to t-code SE24 and create a class

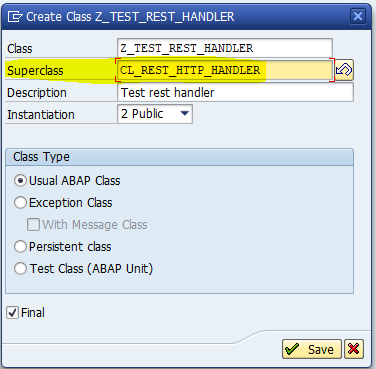




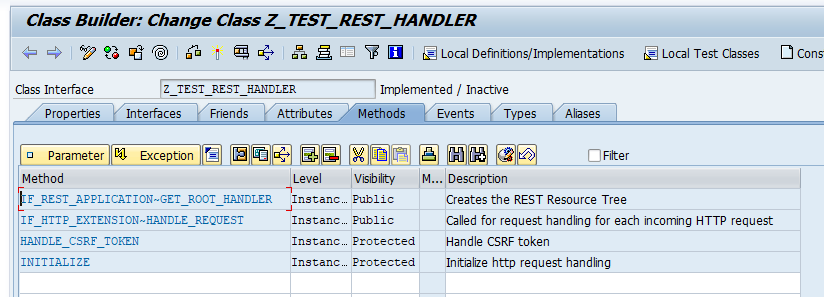
1. On the next pop-up screen provide description and click on “Create Inheritance” icon:



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

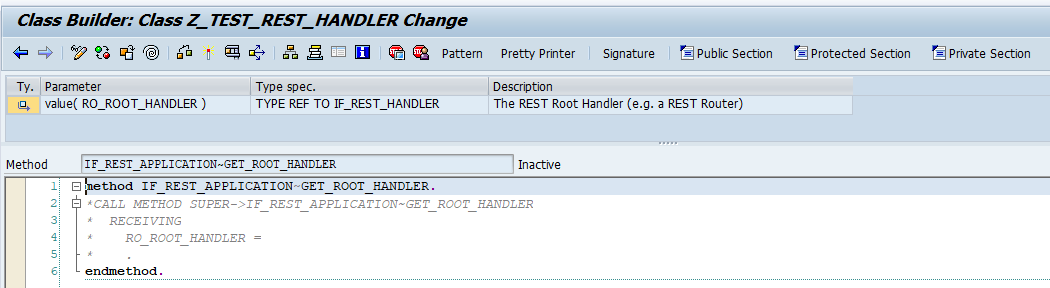


We should be able to see following 4 methods inherited from parent class:



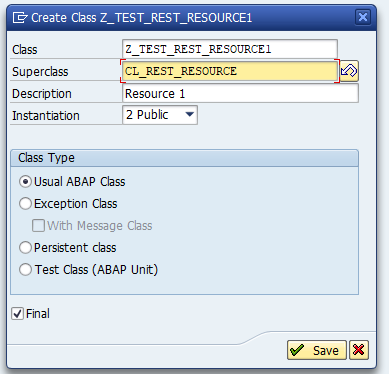
Please note that GET\_ROOT\_HANDLER is an abstract method and hence it is mandatory to redefine the method.

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

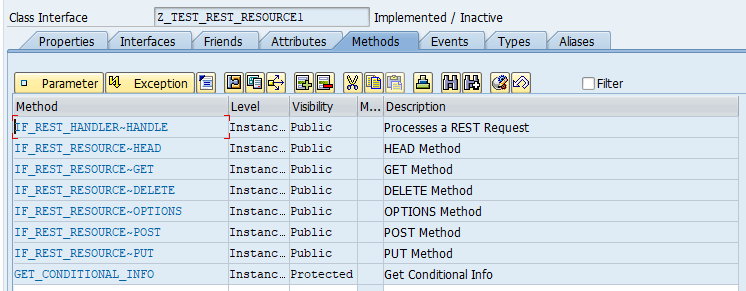


## Create RESOURCE class:

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

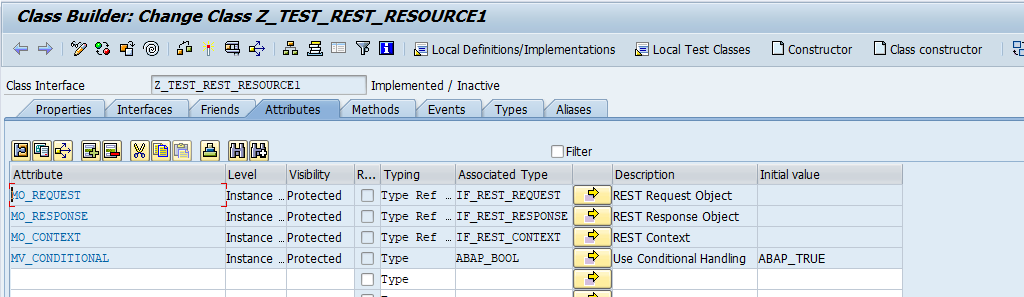


On save, we should be able to see following methods:

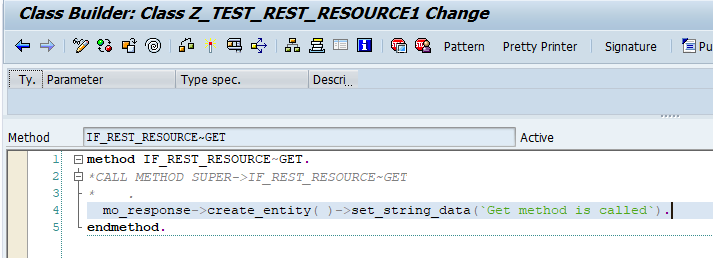


Unlike CL\_HTTP\_REST\_HANDLER, all these methods are implemented in parent class where each of these methods raise an exception “Not allowed” by default. This ensures that only those methods are valid which are implemented by us(developer).

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.

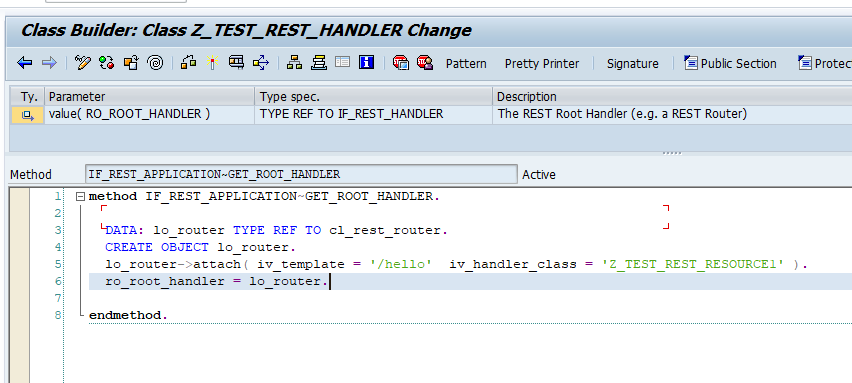


Let us re-define GET method and POST method with a simple message as response body:



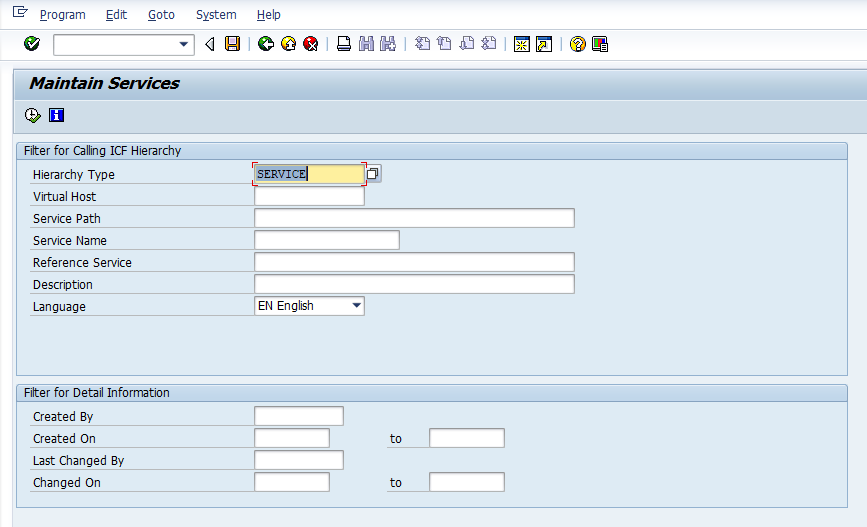
## Attach the RESOURCE class to a route in HANDLER class:

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

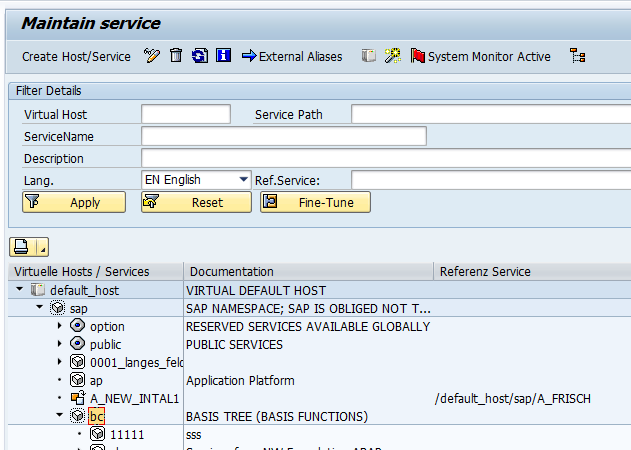


## Create ICF service

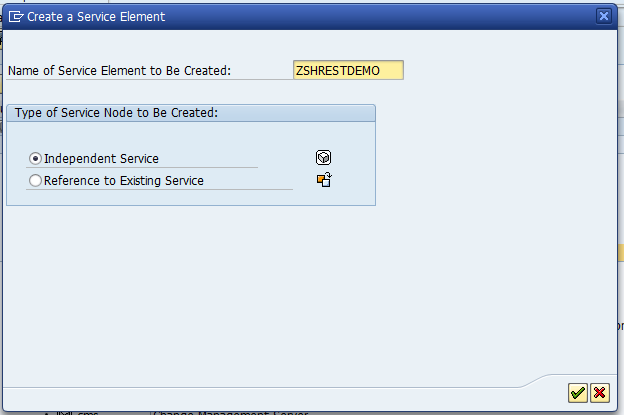
1. Run t-code SICF



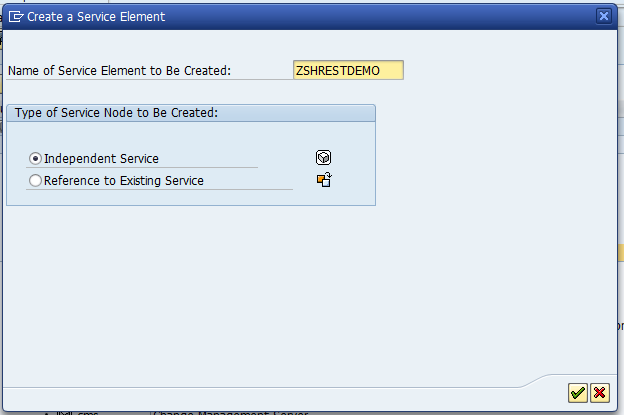
1. Execute with Hierarchy type as “SERVICE”
2. Navigate to default\_host/sap/bc



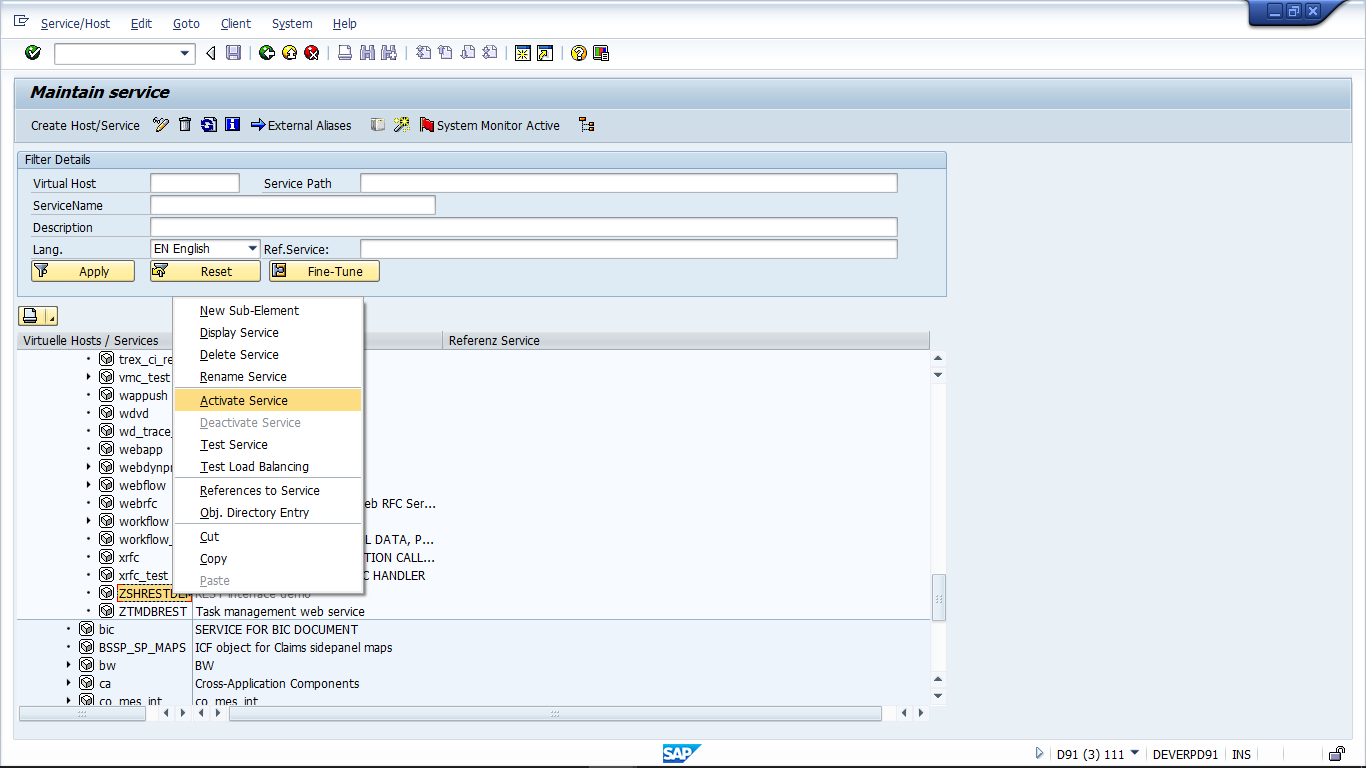
1. Right click on “bc” and click on “New sub-element”
2. Click ok on information message window
3. Enter name of Service and select radio button “Independent Service”



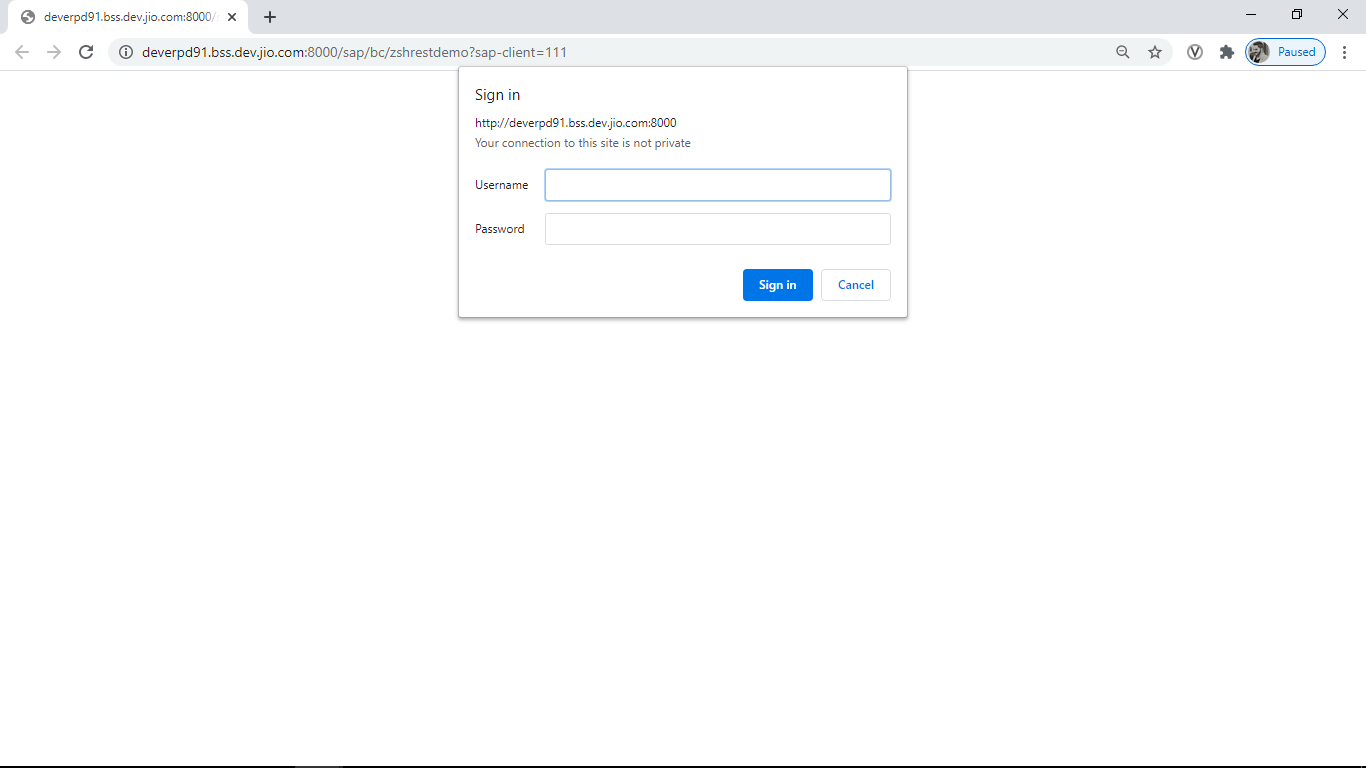
1. Right click on “bc” and click on “New sub-element”
2. Click ok on information message window
3. Enter name of Service and select radio button “Independent Service”

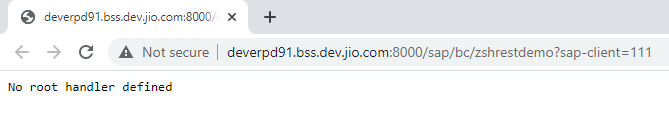


1. Enter implemented HANDLER class name in Handlers List
2. Click on back and activate created sub-element



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 “ “.

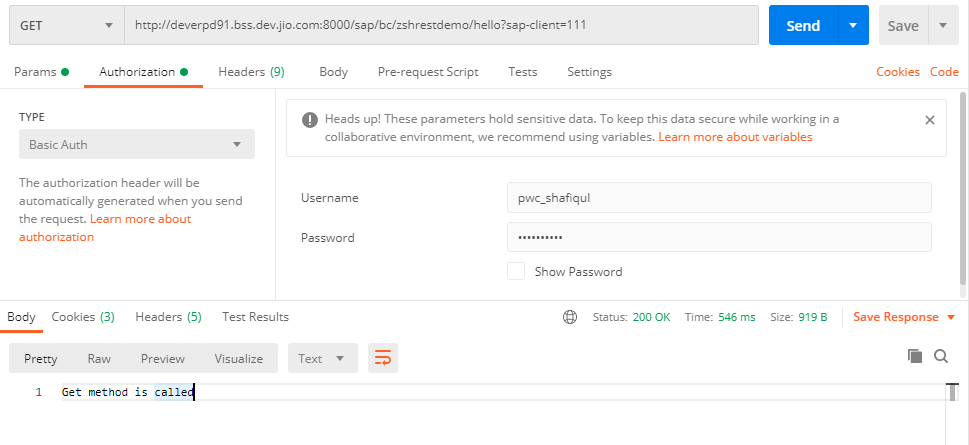
However, we have mapped route “/hello” to a resource class so let us try that route.

## 

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

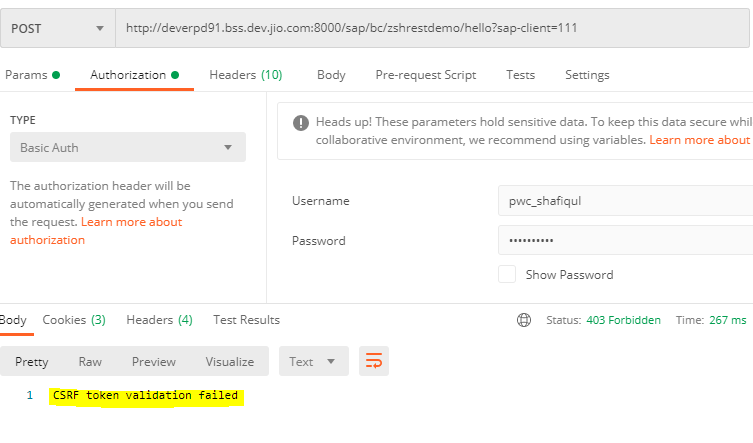
## TEST API using POSTMAN

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



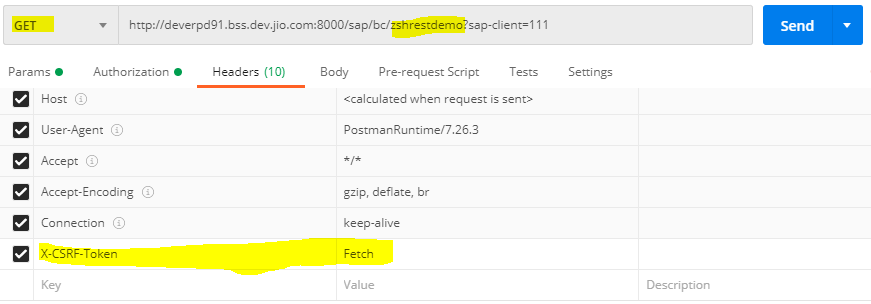
The method is called successfully.

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

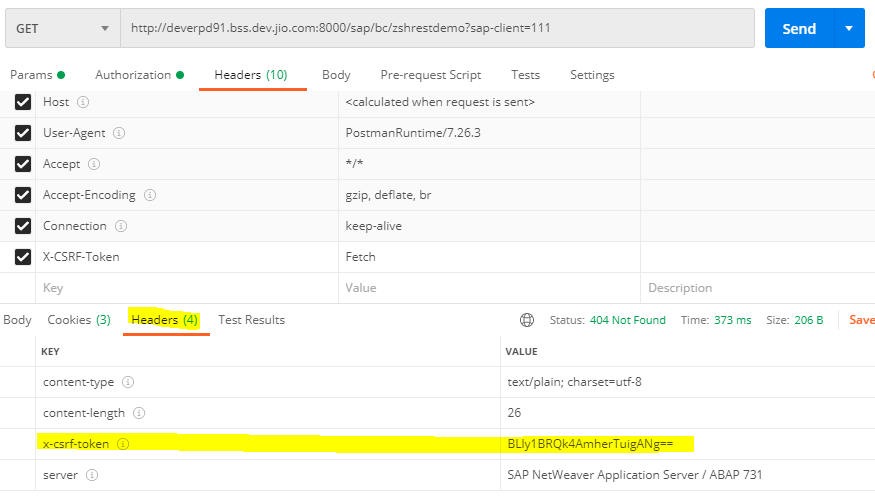


As we can see, POST call failed with error “CSRF token validation failed”.

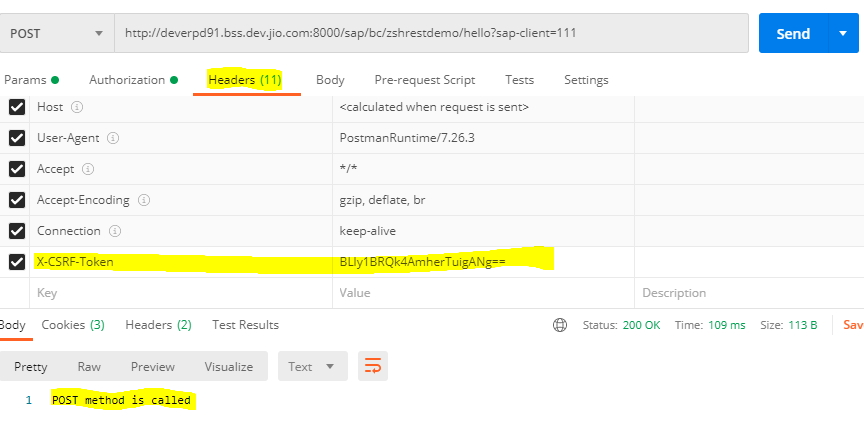
To call POST/PUT/DELETE method, we need to first fetch a valid X-CSRF token from this endpoint or any valid endpoint of the server using GET method and header field “{X-CSRF-Token:Fetch}”.



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.

## Conclusion

This document provide details on how to create a working model of REST API in SAP. However, handling of body part of the requests and response needs further detailing on JSON/XML, serializing/deserializing of such data and some more details to have a complete understanding of REST API for leveraging the sound features REST API provides.