SFDC Rest APIs

There are two types of Rest APIs in Salesforce

1. Standard Rest APIs
2. Custom Rest APIs

Both these APIs support both standard as well as custom objects so don't confuse with their names.

Q: How to create a standard Rest API

Step 1: create connected app in Salesforce

Go to setup and quick search apps. Under connected apps create new app. Make sure you do **Enable OAuth Settings** and provide relevant access. Save and note down client id and secret.

For calling an API you need bearer token. Your client application will generate the bearer by using following details. You can also use Postman to generate the same.

Auth URL https://login.salesforce.com/services/oauth2/token

Post this data to above URL: grant_type=password&client_id=*client_id*&client_secret=*secret*&username=*your salesforce username*&password=*your salesforce pwd followed by password security token*

Above URL will return bearer token which you can use in your client application or in Postman.

Step 2 Create API

2.1 Creating and using standard Rest API

You really don't need to do anything to create a standard API. It is automatically available in Salesforce for a given object. E.g. following is the URL for custom object Employee__c

https://anil99-dev-ed.my.salesforce.com/services/data/v45.0/composite/tree/Employee__c

Note: make sure you use your own domain name and object name in URL above.

Request JSON format for inserting records (request type: post):

Insert single record	**Insert multiple records**

{
"Name" : "Anil Sharma",
"Other_Field__c" : "other field value",
}
	{
	  "records": [{
	    "Name": "Anil Sharma ",
	    "Other_Field__c": "Other field value ",
	
	    "attributes": {
	      "type": "Employee__c",
	      "referenceId": "ref1"
	    }
	  }, {
	    "Name": "Krishna Gupta",
	    "Other_Field__c": "Other field value ",
	    "attributes": {
	      "type": "Employee__c ",
	      "referenceId": "ref2"
	    }
	  }]
	}

2.2 Creating and using custom Rest API

Custom rest API is an Apex class with rest annotation e.g.

```
@RestResource(urlMapping='/MyAPI/*')
global with sharing class MyRestResource {

  @HttpDelete
  global static void doDelete() {
    RestRequest req = RestContext.request;
    RestResponse res = RestContext.response;
    String accountId = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
    Account account = [SELECT Id FROM Account WHERE Id = :accountId];
    delete account;
  }

  @HttpGet
  global static Account doGet() {
    RestRequest req = RestContext.request;
    RestResponse res = RestContext.response;
    String accountId = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
    Account result = [SELECT Id, Name, Phone, Website FROM Account WHERE Id = :accountId];
    return result;
  }

 @HttpPost
  global static String doPost(String name,  String phone, String website) {
    Account account = new Account();
    account.Name = name;
    account.phone = phone;
    account.website = website;
    insert account;
    return account.Id;
  }
}
```

Sample JSONs for above three requests:

GET
URL https://<your domain>.salesforce.com/services/apexrest/MyAPI/accountID replace account id with actual id

POST
URL https://<your domain>.salesforce.com/services/apexrest/MyAPI Request JSON {“name” : “anil sharma”,“phone” : “1234567890”,“website” : “www.example.com”, }

DELETE
URL https://<your domain>.salesforce.com/services/apexrest/MyAPI/accountID replace account id with actual id

Refresh token

```
POST /services/oath2/token HTTP/1.1
Host: login.salesforce.com
Authorization:  Basic
client_id=3MVG9lKcPoNINVBIPJjdw1J9LLM82HnFVVX19KY1uA5mu0
QqEWhqKpoW3svG3XHrXDiCQjK1mdgAvhCscA9GE&client_secret=1955279925675241571

Response: grant_type=refresh_token&refresh_token=your token here
```

Session timeout

Timeout settings in connected app or user profile or org profile

- The session timeout for an access token can be configured in Salesforce from Setup by entering `Session Settings` in the Quick Find box, then selecting **Session Settings**.
- If the application uses the username-password OAuth authentication flow, no refresh token is issued, as the user cannot authorize the application in this flow. If the access token expires, the application using username-password OAuth flow must reauthenticate the user.