FREE eBook

LEARNING sugarcrm

Free unaffiliated eBook created from
Stack Overflow contributors.

#sugarcrm

Table of Contents

About 1

Chapter 1: Getting started with sugarcrm 2

Remarks 2

Examples 2

Installation or Setup 2

Chapter 2: v10 REST API 3

Introduction 3

Examples 3

Logging in 3

Basic CRUD operations 4

Creating custom Endpoints 5

Credits 8

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: sugarcrm

It is an unofficial and free sugarcrm ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sugarcrm.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with sugarcrm

Remarks

This section provides an overview of what sugarcrm is, and why a developer might want to use it.

It should also mention any large subjects within sugarcrm, and link out to the related topics. Since the Documentation for sugarcrm is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting sugarcrm set up or installed.

Read Getting started with sugarcrm online: https://riptutorial.com/sugarcrm/topic/10038/getting-started-with-sugarcrm

Chapter 2: v10 REST API

Introduction

SugarCRM's RESTful API was updated to version 10 in the SugarCRM 7.x releases.

As the 7.x versions are built as a single page app, the API is the main means of communication with the server to retrieve data. Third party integrations also can tie in and access the data. The API can also be extended to provide custom endpoints, implementing desired functionality.

Official docs

Examples

Logging in

You can use this example as a call from code, or through a REST client such as Postman:

```
POST https://YOURSITE.com/rest/v10/oauth2/token
```

```
{
"grant_type":"password",
"client_id":"sugar",
"client_secret": "",
"username":"your_username",
"password":"your_P@$sw0rd",
"platform":"api"
}
```

To elaborate on the above details, they stand for:

> **"grant_type":"password"** - This is the method of logging in. In this case, we're using a username/password combination, so we put through "password"
>
> **"client_id":"sugar"** - This indicates that we are authenticating ourselves through the standard "sugar" client of name/password. Other clients such as "support_portal". This also can reference a custom OAuth key within the application.
>
> **"client_secret": ""** - This is blank as we're using the client id of "sugar", but if you're using a custom client, this is the secret associated with the client.
>
> **"username":"your_username"** - Your user name
>
> **"password":"your_P@$sw0rd"** - Your password
>
> **"platform":"api"** - The typical platforms used within sugar are "base", "mobile", and "portal". For security reasons, if the same user logs into the same platform simultaneously, it will log the user out of the previous session. I've used "api", but this

could be "myawesomesugarintegration" if you wanted.

Upon a successful request, the server returns an object including an "access_token", i.e:

"access_token":"**abcdef01-2345-6789-0abc-def012345678**"

For each subsequent request to the API, you must include this token to authenticate yourself.

That can be achieved by adding the header `"OAuth-Token"` with the access_token value to any further requests.

Basic CRUD operations

First, ensure you log in, and add the access_token to your request header.

In this example, we're going to do some basic operations to access records. These use the Accounts module as an example, but other standard and custom modules (i.e. Leads, Contacts, Opportunities) behave in the same way.

Create

To create an Account, we need to make a post request to the Accounts endpoint, with the details we want to add. Upon success, this returns an object containing the new record's ID, and the current data.

```
POST https://YOURSITE.com/rest/v10/Accounts
{
"name":"My New Account"
}
```

Read

First, we'll retrieve the record we just created, for me the Id was "9174c58c-409c-11e7-bfdf-00163ef1f82f" so to retrieve all information for the record, we do the following:

```
GET https://YOURSITE.com/rest/v10/Accounts/9174c58c-409c-11e7-bfdf-00163ef1f82f
```

That sure is a large object! What about just seeing the name, and the date I just created it?

```
GET https://YOURSITE.com/rest/v10/Accounts/9174c58c-409c-11e7-bfdf-
00163ef1f82f?fields=name,date_entered
```

Much better. But what if I have thousands of Accounts in the system already, and haven't managed to remember the GUID for it?

```
GET https://YOURSITE.com/rest/v10/Accounts?fields=name,date_entered&filter[0][name]=My New
Account
```

Update

So what if I decided I wanted to change something on the app? What about changing the name, and adding a description?

```
PUT https://YOURSITE.com/rest/v10/Accounts/9174c58c-409c-11e7-bfdf-00163ef1f82f
{
    "name":"My Updated Account",
    "description":"Updated via REST API"
}
```

Delete

Okay, that's all fine, but let's clear this up before putting real data in:

```
DELETE https://YOURSITE.com/rest/v10/Accounts/9174c58c-409c-11e7-bfdf-00163ef1f82f
```

Creating custom Endpoints

One of the features in SugarCRM 7.x is being able to easily add and extend custom endpoints to accomplish what you require.

In this example, we'll create a couple of custom endpoints to return some data about the request.

This custom file is being placed in `custom/clients/base/api/DescriptionAPI.php`.

```
<?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point');

//You need to inherit the SugarApi Class to
class DescriptionApi extends SugarApi
{
    public function registerApiRest()
    {
        return array(

            //Define a key for the array
            'DescribeRequest' => array(

                //Array of the acceptable kinds of requests for this method
                'reqType' => array('GET','POST','PUT','DELETE'),

                //If true, anyone can access. If false, only authenticated users.
                'noLoginRequired' => true,

                //Here is the path to access the endpoint, in this case: Describe/Request
                'path' => array('Describe', 'Request'),

                //Specify an empty string for the path variables
                'pathVars' => array('', ''),

                //method to call
                'method' => 'DescribeMyRequest',

                //A small description, displayed in rest/v10/help page
                'shortHelp' => 'Describes your Request Method',
```

```
                //Further help, displayed when drilling down into the help page
                'longHelp' => 'custom/clients/base/api/help/DescribeRequestHelp.html',
            ),
            //Here's another entry with some more in depth information
            'DescribeIncludingArgs' => array(
                'reqType' => array('GET','POST','PUT','DELETE'),
                'noLoginRequired' => true,

                //This time, we'll include a third element with a ?
                //So now the path is Describe/Request/{dataFromURL}
                'path' => array('Describe', 'Request', '?'),

                //Here, we specify the key for accessing that data within the function
                'pathVars' => array('', '','dataFromURL'),

                'method' => 'DescribeMyRequestIncludingArguments',
                'shortHelp' => 'Describes the request you sent, including method, URL
parameters, and request body',
                'longHelp' => 'custom/clients/base/api/help/DescribeIncludingArgsHelp.html',
            ),
        );

    }

    /**
     * Your custom logic goes in here.
     */
    public function DescribeMyRequest($api, $args)
    {
        //Find out the request method sent
        $requestType = $_SERVER['REQUEST_METHOD'];

        return "You sent a $requestType request.";
    }

    /**
     * Here is the second function
     */
    public function DescribeMyRequestIncludingArguments($api, $args)
    {
        //Find out the request method sent
        $requestType = $_SERVER['REQUEST_METHOD'];

        //Get the data included in the URL parameter
        $data = $args['dataFromURL'];

        //Read from the request body
        $body = file_get_contents('php://input');

        return "You sent a $requestType request including the header argument: `$data` and the
body: `$body`";
    }

}
```

After adding this file, you'll need to perform a Repair and Rebuild in order for Sugar to register your endpoint correctly.

After, if you navigate to rest/v10/Describe/Request in your browser, you should see:

> "You sent a GET request."

Now, if you use a REST client to send a POST request, with some data, for example: `POST rest/v10/Describe/Request/Stuff` with the body `{"key":"value"}` you should receive:

> "You sent a POST request including the header argument: `Stuff` and the body: `{\"key\":\"value\"}`"

Read v10 REST API online: https://riptutorial.com/sugarcrm/topic/10043/v10-rest-api

Credits

S. No	Chapters	Contributors
1	Getting started with sugarcrm	Community
2	v10 REST API	Reisclef