MOMENTUM 2016

HACK-A-THON

DOCUMENTUM REST API LAB

Table of Contents

Overview

Welcome to Documentum REST API lab

This EMC lab will walk you through some of the common challenges for developing a responsive front-end web application for common content management functionalities using the REST API at the backend.

In this lab you will learn how EMC Documentum REST Services quickly helps to model your common content management functions in an AngularJS application and gives you the easy access to Documentum content management system with the hypermedia driven API service.

Introduction to Documentum AngularJS File Manager application

In this Hackathon lab, we will build a **Documentum AngularJS File Manager** application. Folder and document operations are the key ability of any enterprise content management system. Documentum AngularJS File Manager is a very simple and intuitive web application which provides the functions to manage Documentum repository folders and contents through a modernized web application. As you get from its name, its front-end mainly leverages AngularJS to build the application. This project's goal is to demonstrate how to use Documentum REST Services to develop modern front-end Single Page Applications.

Angular File Manager

Angular File Manager is a Single Page Application built with AngularJS and Material Design. In this sample project, we leverage the GitHub open source project Angular File Manager to build the main UI components of the folder and document operations. We make customizations for API calls, to make the JavaScript have direct calls to Documentum REST Services. We also added some functions to make the application complete, for instance, the login and logout, the search and so on.

Section 1 Lab Introduction

Documentum REST Services overview

Documentum REST Services is a set of simple yet powerful hypermedia driven REST services which provider API access to Documentum Content Server repositories. Documentum REST Services supports a lot of service operations, while in this lab, we will only cover a number of common folder and document operations. Documentum REST Services also supports a rich set of authentication schemes, and in this lab, we only use the simplest one, the HTTP Basic authentication. Documentum REST Services supports both XML and JSON formats for its resource representations. JSON is simper in general usage so we consume Documentum REST Services with JSON format in this lab.

Virtual machine

Everyone will a virtual machine for this lab. The virtual machines contains all the software to develop and run the demo project.

Tip

You don't need to install anything on your own laptop, unless you do not have the Remote Desktop software installed yet. All tasks in this lab shall be finished in the virtual machine.

Technology stack

Npm

Node.js is an asynchronous event driven framework designed to build scalable network applications. Node.js has its own package manager to install modules called npm. Npm makes it easy for JavaScript developers to share and reuse code, and it makes it easy to update the code that you're sharing.

A package can be downloaded with the command **npm install <package>**.

Bower

Bower is your front-end package manager which maintains a flat list of dependencies for your front-end applications, placing the burden on you to deal with version conflicts.

Bower is installed by npm with the command **npm install –g bower**. Then you can use bower to install your front-end packages, with the command **bower install <package>**.

Gulp

Gulp is a toolkit to build JavaScript (and other) applications with tasks. It's streaming and code-over-configuration.

Gulp is installed by npm with the command **npm install –g gulp**. Then you can use gulp to build your front-end application with tasks, using the command **gulp <task>**.

AngularJS

AngularJS is a JavaScript Model-View-Whatever (MVW) framework. The AngularJS framework works by first reading the HTML page, which has embedded into it additional custom tag attributes. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables, then can be managed by JavaScript codes.

Development approach in this lab

The general approach we take when building AngularJS applications is to start with some static HTML and then gradually add behavior. Our main purpose for this course is to learn how to consume Documentum REST Services in JavaScript clients, so we will have more focus on the API calls.

The hackathon will consist of a number of rounds of coding. After each round, we'll stop coding and take some time to reflect on what we've done so far. If you haven't made as much progress as you would have liked in a round, you can catch up in between rounds, because we provide all the code in GitHub.

For tasks that require your coding actions, you will see this hint before the code "Your code at line xxx".

EMC Documentum @GitHub

GitHub is the world's largest code host. Built on the powerful Git version control system, GitHub is the ideal place to share code.

Code in GitHub lives in repositories. A repository contains all of the project files (including documentation), and stores each file's revision history. Repositories can have multiple collaborators and can be either public or private. The public GitHub repository for this hackathon is located at

https://github.com/Enterprise-Content-Management/documentum-rest-client-angular-filemanager

Branching is the duplication of source code under version control so that modifications can happen in parallel. Here we'll use branches a bit differently. Each round of coding has its own branch, so that you can start each round on the same page as everybody else. You get the code in a branch using the command

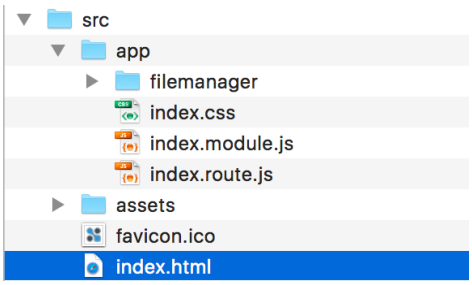
git checkout -B <branch_name> origin/<branch_name> –f

Information

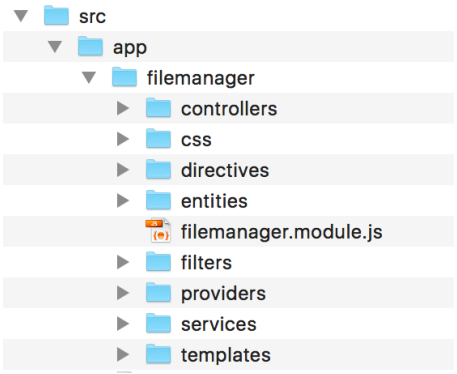
To learn more Documentum REST clients, please visit http://enterprise-content-management.github.io/rest/.

Code Overview

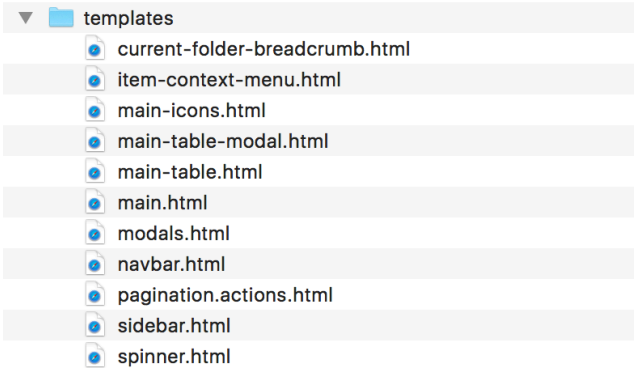
This code uses a structure often found in moderately complex Angular applications. The main index page is *src/index.html*, which is essentially an empty shell into which Angular injects the components that make up the program itself. The components themselves are found in *src/app* and its subdirectories. The file *src/app/index.route.js* contains the configuration that determines which components are injected into *src/index.html*.



The *src/app/filemanager* folder contains all the Angular components in the application.

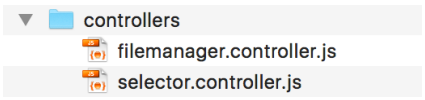


The templates folder contains the HTML templates that define the main screen and the various components that appear on it.



main.html defines the screen as a whole. It includes other templates such as *navbar.html,* and *sidebar.html.*

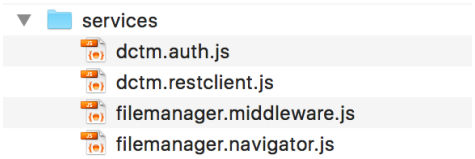
The *filemanager/controllers* directory contains the Angular controllers:



filemanager.controller.js is the main controller for the app. It handles events like *login, logout, getRepositoryList*, etc., calling Angular services to invoke Documentum services.

selector.controller.js is used to handle selection of items in the filemanager.

The *filemanager/services* directory contains the Angular services that invoke the Documentum REST services:



All RESTful services used for authentication are invoked in *dctm.auth.js*. All RESTful services used for speaking to Documentum per se are invoked in *dctm.restclient.js.*

Section 2 File Manager Development

Round 0: Project setup

In the first round, we will check the Documentum software set up and build the initial project in the virtual machine. You shall now need to remote desktop connect to your virtual machine.

Documentum Content Server

Content Server docbroker and repositories are started up by default. If not, please open Documentum Manager to start them. There is only one repository installed in this demo environment, whose name is repo1.

Documentum REST Server

Documentum REST Services 7.3 build is deployed to the Tomcat 7 server at C:\Tomcat7i2\webapps\dctm-rest. It's started up by default. If not, please double click the startAll.bat on your VM's desktop or navigate to C:\Tomcat7i2\bin\startup.bat start it up. The HTTP port for the REST services is 8083, so the welcome page for Documentum REST Services is http://localhost:8083/dctm-rest. You can have a try in the web browser.

If you deploy Documentum REST server from scratch, you need to update below two files by yourself.

dctm-rest.war\WEB-INF\classes\dfc.properties

dfc.docbroker.host[0]= <fill the value>
dfc.docbroker.port[0]= <fill the value>
dfc.globalregistry.repository= <fill the value>
dfc.globalregistry.username= <fill the value>
dfc.globalregistry.password= <fill the value>

dctm-rest.war\WEB-INF\classes\rest-api-runtime.properties

rest.cors.enabled=true

Documentum Administrator

Documentum Administrator (DA) is deployed to the same Tomcat server, so now you are able to navigate Content Server repository by DA, too. It's root URL is: http://localhost:8083/da.

For any application that requires authentication to the Content Server repository, you can always use the username dmadmin and password D3m04doc!.

Get the code

Each round in the lab you can either build on your code from the previous round or you can simply clone the branch for that round from GitHub. To get started you should clone the branch for the first round, which is labeled **mmtm-r1**:

1. Open a command window
2. Browse to: **C:\momentum**

3. Run the command: **git clone https://github.com/Enterprise-Content-Management/documentum-rest-client-angular-filemanager.git -b mmtm-r1**
4. Now the project shall be downloaded to location **C:\momentum\documentum-rest-client-angular-filemanager**

Tip

Just for shortening in the below paragraph, we will use ***<root>*** *to represent the file location* ***C:\momentum\documentum-rest-client-angular-filemanager.***

Build from CLI

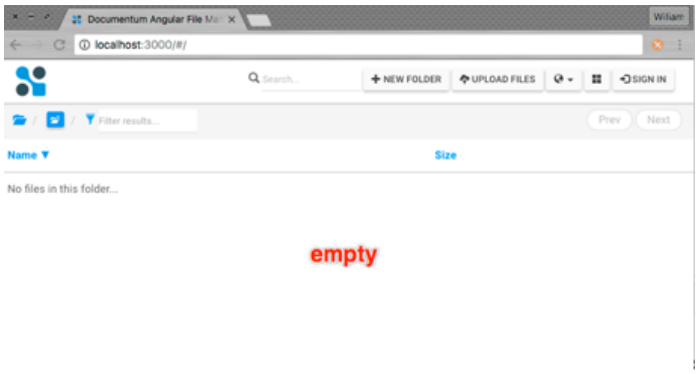
The project is built with Npm, Bower and Gulp. The build environment has been set up already so we can just start for build.

Open a new Command-line window from current directory **<root>**.

1. Run the command **npm install –save-dev** to install all dependencies listed in package.json, which contains node modules.

   > ***Tip****: If npm runs into errors, please try* ***npm cache clean*** *and install again. If it does not work, you can try to manually remove folder* ***C:\Users\dmadmin\AppData\Roaming\npm-cache***

2. After it finishes, run the command **bower install** to install all dependencies listed in bower.json, which contains Javascript libraries used in the application.
3. After it finishes, run the command **gulp**, a build system that can also launch a web server to run an app locally. The build tasks are defined in **gulpfile.js.**
4. After it finishes, run the command **gulp serve** to serve the app in the default web browser at **http://localhost:3000/#**



By completing the above steps, the project File Manager has been built and run successfully. But you can do nothing with it because functions are not implemented yet.

Import into IDE

As we will modify the project to add missing functions, we need to import the project into an IDE. In this lab, we will use Visual Studio Code (VSC), which you can see on your destop.

1. Open VSC.
2. Click on **File** > **Open Folder…** >
3. Browse to the location of the project: **<root>**
4. Click on **Open**. The project is loaded to VSC. You can browse the project structure by the navigator in left side bar.

Now let's take a few minutes to navigate the project structure.

Tip

One big advantage of Gulp build is that once you save your work for the modified source files, the running application will be refreshed automatically on the same port. It makes your changes visible in the UI immediately.

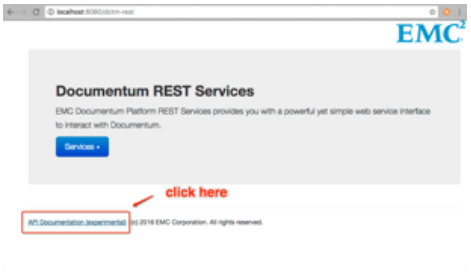
Validating the environment

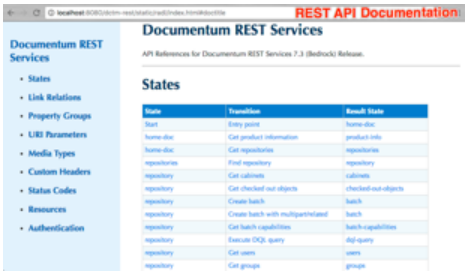
Documentum REST server

Jonathan Robie 10/27/2016 10:29 PM
Comment [1]: I was confused by these headers. How are they to be understood?

To do parallel comparison, you can always check the folders and documents from DA, what you can do in the File Manager you can also do in DA.

On your Documentum REST Services port, you can also view the REST API documentation at http://localhost:8083/dctm-rest/static/radl/index.html.

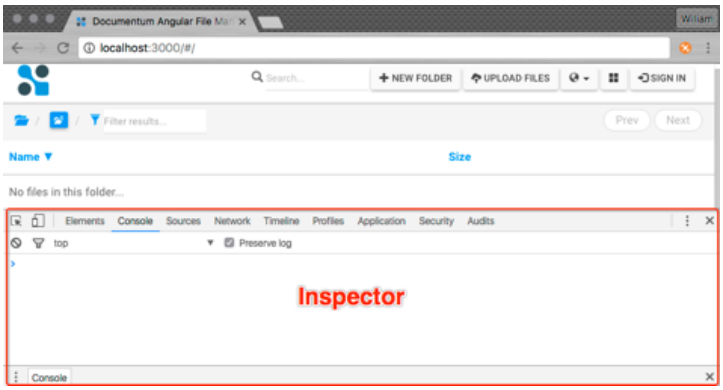




file-manager

For the Angular File Manager project, you can use the browser's dev tool to debug the JS code. Here is the sample for Chrome web browser.

1. **Right Click** on the page of File manager app, and click on **Inspect**.
2. We can observe every piece of the HTML/JS elements from the inspector. In this lab, we mainly watch on **Console**, **Sources** and **Network** tabs. Please see samples below.



If necessary, we can add debug breakpoints on the project code to run the Java Script code step by step.

Summary

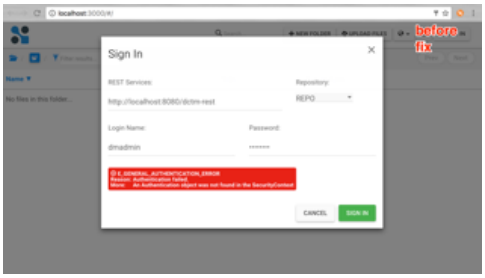
In this round, we get the code for Documentum AngularJS File Manager, build and run it.

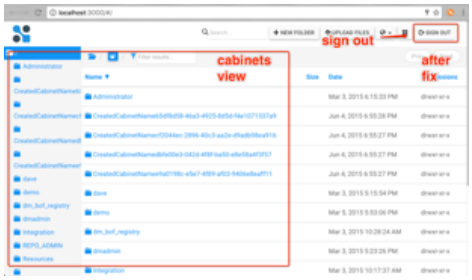
- Command to start File Manager: **<root>\gulp serve**
- File Manager port: **http://localhost:3000**

Round 1: Sign-in to repository

Overview

In this round, we will learn how to make an authenticated REST request. On opening the File Manager UI, you can find a **sign-in** button on the right top menu. By now, your login will fail. Here are the diagrams showing the difference before and after the fix.





Get the code

We retrieved the code base in round 0, so at this moment you don't need to re-fetch the code from server. If you haven't downloaded the code, please refer to Get the code.

Practice

Now let's examine where is the code for login.

(1) Open the file **<root>/src/app/filemanager/templates/modals.html**, which defines modal dialogs for the application. You will find a directive for **sign in**. This directive calls **login()** method from filemanager.controller.js.

```
<div class="modal animated fadeIn" id="signin">
  <div class="modal-dialog">
    <div class="modal-content">
        <form ng-submit="login()">
        …
```

(2) Open source code **<root>/src/app/filemanager/controllers/filemanager.controller.js**. You will find the function **login()**. It calls the **filemanager.middleware.js** and then **dctm.restclient.js** to login to a repository.

```
$scope.login = function() {
    $scope.apiMiddleware.login().then(function() {
        $scope.fileNavigator.refresh();
        $scope.modal('signin', true);
    });
};
```

(3) Open source code **<root>/src/app/filemanager/services/dctm.restclient.js**. You will find the function **login()**. It calls **dctm.auth.js** to attempt a login.

```
dctmRestClient.prototype.login = function(username, password, repositoryUrl) {
    var self = this;
    dctmAuth.try_login(username, password);
    return $http.get(repositoryUrl).success(function(data) {
        dctmAuth.post_login(username, password, data);
    })
    .error(function(data) {
        parseError(self, data, 'error_authenticating');
        dctmAuth.logout();
    });
}
```

Jonathan Robie 10/27/2016 10:34 PM

Comment [2]: Multiple methods are called login(). How do I know which one this refers to?

William Zhou 10/28/2016 5:16 PM

Comment [3]: As said here, from **filemanager.controller.js**

Jonathan Robie 10/27/2016 10:39 PM

Comment [4]: This description is not very clear. It looks like $scope.login is set to an anonymous function that calls apiMiddleware.login, refreshes the file navigator, and invokes the modal mentioned in (1)? Or am I misunderstanding this?

William Zhou 10/28/2016 5:18 PM

Comment [5]: It's async invocation mode in javascrit
1.Call apimiddleware.login
2.After success -> refresh the file navigator, and mark the modal state (not invoke).

Jonathan Robie 10/27/2016 10:44 PM

Comment [6]: This is not at all clear about how the various functions are invoked.

What is the relationship between:
- •service.try.login
- •dctmRestClient.prototype.login
- •$scope.login

How is this relationship established?

And isn't this the function item assigned to dctmRestClient.prototype.login, as opposed to "the login() function"?

William Zhou 10/28/2016 5:23 PM

Comment [7]: dctmRest.Client.login does a real http login (http call);
dctmAuth provides js in-process API calls, no actual http calls.
Here the logic for login is:
- try_login → set request auth header
- send http call for repository url with the header
- after success → save the header for all subsequent requests
- after failure → clear header

(4) Open source code **<root>/src/app/filemanager/services/dctm.auth.js**. You need to implement the login now.

Your code at line 10 – complete the folder list call for a non root folder by calling a function from middleware.js.

```
// TODO for MMTM R1: set authentication for http START
    service.try_login = function (username, password) {
    // set authentication for http
};
// TODO for MMTM R1: set authentication for http END
```

a

Code hint

a. The default REST deployment supports HTTP Basic authentication.

Answer

```
service.try_login = function (username, password) {
    var authdata = Base64.encode(username + ':' + password);
    $http.defaults.headers.common['Authorization'] = 'Basic ' + authdata;
};
```

Code explanation

a. HTTP Basic authentication requires username and password encrypted by BASE64.

Now save your work, refresh the page, and login again.

Summary

- File Manage The default REST deployment supports HTTP Basic authentication.

Jonathan Robie 10/27/2016 10:44 PM
Comment [8]: If you didn't tell me what file to open, how would I figure this out for myself in Visual Studio?

William Zhou 10/28/2016 5:21 PM
Comment [9]: See step (2), **dctmAuth**.try_login, it's a func referencing from this js file. Please note the module name declaration at the top of the js file.

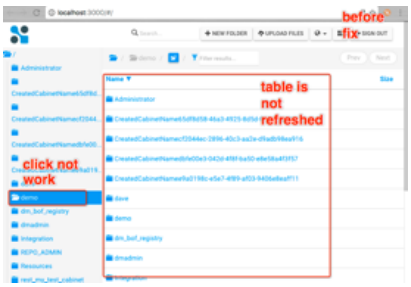
Jonathan Robie 10/27/2016 11:08 PM
Comment [10]: As written, this implies that the target user would be able to look at the TODO and the Code Hint, then write this code. Do we expect workshop participants to be able to do that? If not, we might make it plain that it's OK to look at the answer and copy it in.

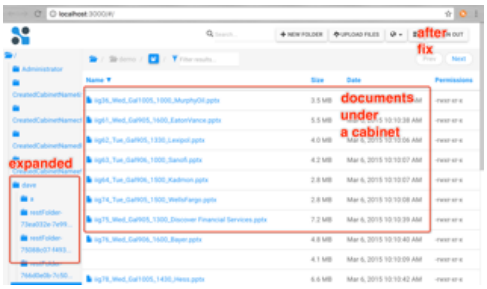
William Zhou 10/28/2016 5:26 PM
Comment [11]: Yes, they can read it (not good enough but docx doesn't allows us to hide the text without extensions)

Round 2: Folder navigation

Overview

In this round, we will learn how to list cabinets and folders under a repository. Now after you sign into the repository you are able to list the cabinets, but you are not able to navigate down from the cabinet. Here are the diagrams showing the differences before and after the fix.





Get the code

1. Open a Command window
2. Browse to **<root>**
3. Run: **git checkout –b mmtm-r2 origin/mmtm-r2 –f**
   (note: this will override any changes you have already made)
4. Refresh your project in VSC

Practice

(1) Open source code **<root>/src/app/filemanager/controllers/filemanager.controller.js**
This is the main **AngularJS controller** for the app. At the bottom you will see this code:

```
$scope.fileNavigator.refresh();
```

It indicates that an AngularJS service **filenavigator** takes responsibility to refresh the page. **No actions are required for step (1).**

(2) Open source code Open source code
<root>/src/app/filemanager/services/filemanager.navigator.js
Locate to the function **refresh()**. You'll see that this function builds the folder tree based on the current path. It's incomplete because when the path is not root, the navigation stops.

Your code at line 25 – complete the folder list call for a non root folder by calling a function from middleware.js.

```
FileNavigator.prototype.refresh = function() {
    var self = this;
    var path = self.currentPath.join('/');
    self.list(path, self.folderId, self.folderObject).then(function(data) {
        var objects = self.apiMiddleware.parseEntries(data);
        self.fileList = (objects || []).map(function(file) {
            return new Item(file, self.currentPath);
        });
        self.buildTree(path);
    });
};

FileNavigator.prototype.list = function(path, id, folder) {
    if (path == "/" || path == "") {
        return this.apiMiddleware.listRootCabinets(this.pageNumber, this.pageSize);
    }
    else {
        // TODO for MMTM R2: improve folder list rest api call START
        // call apiMiddleware to get folder children for folder
        // TODO for MMTM R2: improve folder list rest api call START
    }
};
```

a

Code hint

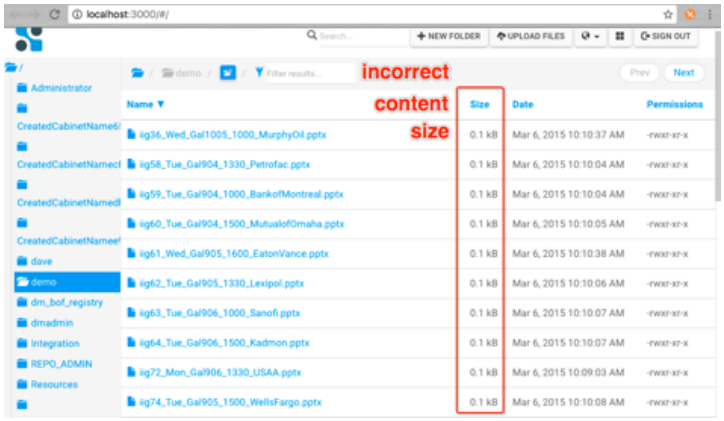
b. The middleware.js is a bridge between the AngularJS and REST client http call. You will be able to find a function to list folder children.

Answer

```
FileNavigator.prototype.list = function(path, id, folder) {
    if (path == "/" || path == "") {
        return this.apiMiddleware.listRootCabinets(this.pageNumber, this.pageSize);
    }
    else {
        return this.apiMiddleware.listFolderChildren(object, this.pageNumber,
this.pageSize);
    }
};
```

Now save your work and refresh the page, you will be able to click on a sidebar cabinet to navigate down to its sub folders and documents. However, there are 2 small issues:

- The folder children lists 100 items by default, which is exceeding one page view in the browser. We want to limit the page size to 20.
- The documents in the folder children table have incorrect content size.



(3) Open source code **<root>/src/app/filemanager/services/dctm.restclient.js**.

Your code at line 65 – add query parameters to customize the pagination and attribute view for folder children list.

```
// TODO for MMTM R2: improve folder list rest api call START
dctmRestClient.prototype.listFolderChildren = function(folder, pageNumber,
itemsPerPage) {
    var requestUrl = this.findUrlGivenLinkRelation(folder,
'http://identifiers.emc.com/linkrel/objects');
    //add query parameters for page size and attribute list
    return get(this, requestUrl, 'error_getting_folder_children');
}
// TODO for MMTM R2: fimprove folder list rest api call END
```

a

Code hint

a. Please read Documentum REST API documentation for **Folder Child Folders Resource**.
- Parameters '**page**' and '**items-per-page**' are used to implement pagination
- Parameter '**view**' is used to customize attribute list in result items

Answer

```
dctmRestClient.prototype.listFolderChildren = function(folder, pageNumber, itemsPerPage) {
    var requestUrl = this.findUrlGivenLinkRelation(folder,
'http://identifiers.emc.com/linkrel/objects');
    var viewAttrs =
'r_object_id,r_object_type,object_name,r_modify_date,r_creation_date,i_folder_id,r_full_co
ntent_size,a_content_type';
    requestUrl = buildUriWithQuery(requestUrl, ['page', pageNumber, 'items-per-page',
itemsPerPage, 'inline', true, 'view', viewAttrs]);
    return get(this, requestUrl, 'error_getting_folder_children');
}
```

a

b

Code explanation

a. Query parameter 'view' allows to specify a list of comma separated attributes.
b. Query parameters 'page' and 'items-per-page' are used to specify a page. Furthermore, the custom attributes view will be returned only when the result items are inline.

Now save your work and refresh the page, you will be able to see the limited page size and content size attributes for documents .

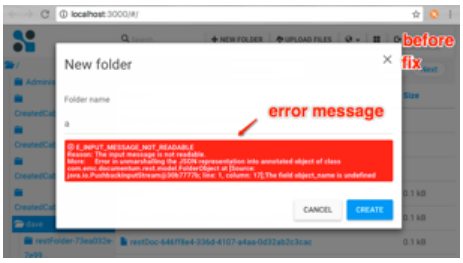
Summary

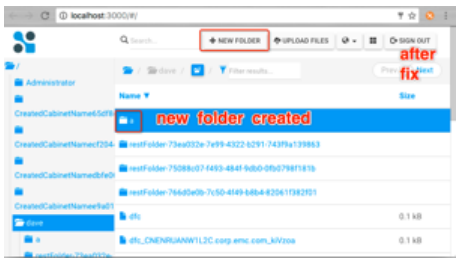
- In Documentum REST Services, some common query parameters are used on collection based resources to customize the result items, including 'view', 'inline', 'page', 'items-per-page' and so on.

Round 3: Create a new folder

Overview

As you see on the UI, there is a button to create new folders on the right top of the page. We will implement this function in this round. Currently, folder creation fails because of invalid REST requests. Here are the diagrams showing the difference before and after the fix.





Get the code

1. Open a Command window
2. Browse to **<root>**
3. Run: **git checkout –b mmtm-r3 origin/mmtm-r3 –f**
   (note: this will override any changes you have already made)
4. Refresh your project in VSC

Practice

We will start with the UI part to see how this is implemented in HTML/JS and what it expects from the server.

(1) Open source code **<root>/src/app/filemanager/templates/navbar.html**. You will notice the HTML tag to create the button for "New Folder".

```
<button class="btn btn-default btn-sm" ng-click="modal('newfolder') &&
prepareNewFolder()">
    <i class="glyphicon glyphicon-plus"></i> {{"new_folder" | translate}}
</button>
```

It indicates that the click on the button triggers to a new modal (template) "**newfolder**". **No actions are required in step (1).**

(2) Open source file **<root>/src/app/filemanager/templates/modals.html**. You will notice this HTML tag with the id "**newfolder**".

```
<div class="modal animated fadeIn" id="newfolder">
  <div class="modal-dialog">
    <div class="modal-content">
        <form ng-submit="createFolder()">
            …
```

The implementation stack for **createFolder()** follows the sequence below: filemanager.controller.js → filemanager.middleware.js → dctm.restclient.js. We are going to fix the issue in dctm.restclient.js. **No actions are required in step (2).**

(3) Open source file **<root>/src/app/filemanager/services/dctm.restclient.js**.

Your code at line 75 – fix the issue of create folder REST api

```
// TODO for MMTM R3: fix create folder rest api call START
dctmRestClient.prototype.createFolder = function(folder, name) {
    var requestUrl = this.findUrlGivenLinkRelation(folder,
'http://identifiers.emc.com/linkrel/folders');
    return post(this, requestUrl, '{\"object_name\":\"' + name + '\"}',
'error_creating_folder');
};
// TODO for MMTM R3: fix create folder rest api call START
```

a

Code explanation

a. Please read Documentum REST API documentation for **Folder Resource** and **Folder Child Folders Resource**.
   - It's right to find Folder Child Folders Resource from Folder Resource's link relation '**http://identifiers.emc.com/linkrel/folders**'
   - It's right to use HTTP method **POST** to create the new folder
   - The request body format is invalid. Please refer to the API documentation JSON sample.

Answer

```
dctmRestClient.prototype.createFolder = function(folder, name) {
    var requestUrl = this.findUrlGivenLinkRelation(folder,
'http://identifiers.emc.com/linkrel/folders');
    return post(this, requestUrl, buildPersistentObject(['object_name', name]),
'error_creating_folder');
};
```

a

Code explanation

a. The right minimal representation for a new folder request body in JSON is:
   {"properties": {"object_name":"xyz"}}

> ***Tip**: you must first click on a folder on the sidebar, then click the button to create new folders. It's because folders in a repository cannot be orphans.*

Now if you save your work, refresh the page, choose a parent folder and create a new sub folder, you will see the successful result.

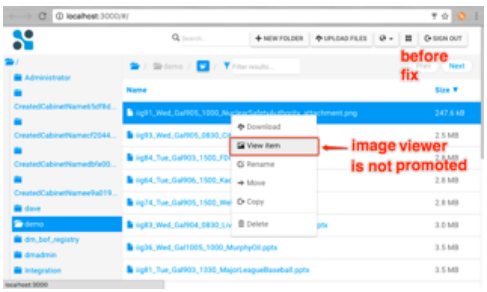
Summary

- In Documentum REST Services, a link relation is defined to discover this resource.
  - folder --> child folders: http://identifiers.emc.com/linkrel/folders
- The HTTP method used to create a new resource is POST.
- The media type for the JSON request body is application/vnd.emc.documentum+json

Round 4: View contents

Overview

In this section, we will learn how to get the content of a document. The file manager supports viewing image content within the browser viewer, so we will **use the image document for testing**. Here are the diagrams showing the difference before and after the fix.



Get the code

1. Open a Command window
2. Browse to **<root>**
3. Run: **git checkout –b mmtm-r4 origin/mmtm-r4 –f**
   (note: this will override any changes you have already made)
4. Refresh your project in VSC

Practice

Firstly, it is still helpful if we take a look at the code to understand what file formats are set as view-able.

(1) Open source code **<root>/src/app/filemanager/providers/config.js**. You can find below code snippet which defines which file formats are view-able.

```
isEditableFilePattern:
/\.(txt|html?|aspx?|ini|pl|py|md|css|js|log|htaccess|htpasswd|json|sql|xml|xslt?|sh|rb
|as|bat|cmd|coffee|php[3-6]?|java|c|cbl|go|h|scala|vb)$/i,
isImageFilePattern: /\.(jpe?g|gif|bmp|png|svg|tiff?)$/i,
isExtractableFilePattern: /\.(gz|tar|rar|g?zip)$/i,
isPdfFilePattern: /\.(pdf)$/i,
```

AngularJS models and views read this config to determine whether specific events are applicable for a selected item. **No actions are required in step (1).**

(2) Open source code **<root>/src/app/filemanager/templates/modals.html**. You can find below code snippet which defines image viewer page.

```
<div class="text-center">
  <img id="imagepreview-target" class="preview" alt="{{singleSelection().model.name}}"
ng-class="{'loading': apiMiddleware.restClient.inprocess}">
  <span class="label label-warning" ng-
show="apiMiddleware.restClient.inprocess">{{'loading' | translate}} ...</span>
</div>
```

The viewer page is controlled by JavaScript and will be shown on **View Item** button. **No actions are required in step (2).**

(3) Open source code **<root>/src/app/filemanager/dctmrestclient.js**. The content media URL for a document is obtained from the **Content (metadata) Resource** in Documentum REST Services, and the Content Resource can be obtained from the **Document Resource**. Given the Document Resource, please implement the API call to get the content metadata.

You action at line103 – implement the content metadata call

```
// TODO for MMTM R4: implement get content metadata rest api call START
dctmRestClient.prototype.getContentMeta = function(docment, distributed) {
    // find content resource uri from document's link relaitons
    // append query parameters if necessary
    // get resoure
    return 'mmtm-r4-content-meta-rest-api-call';
};
// TODO for MMTM R4:  implement get content metadata rest api call END
```

a

Code hint

b. Please read Documentum REST API documentation for Document Resource and Content Resource.
   - Primary Content Resource can be found from Document Resource's link relation '**http://identifiers.emc.com/linkrel/primary-content**'
   - Content Resource supports query parameter '**media-url-policy**'
   - Content Resource supports HTTP method **GET** and **DELETE**

Answer

```
dctmRestClient.prototype.getContentMeta = function(docment, distributed) {
    var requestUrl = this.findUrlGivenLinkRelation(docment,
'http://identifiers.emc.com/linkrel/primary-content');
    requestUrl = buildUriWithQuery(requestUrl, ['media-url-policy', (distributed ?
'DC-PREF':'LOCAL')]);
    return get(this, requestUrl, 'error_downloading_content');
};
```

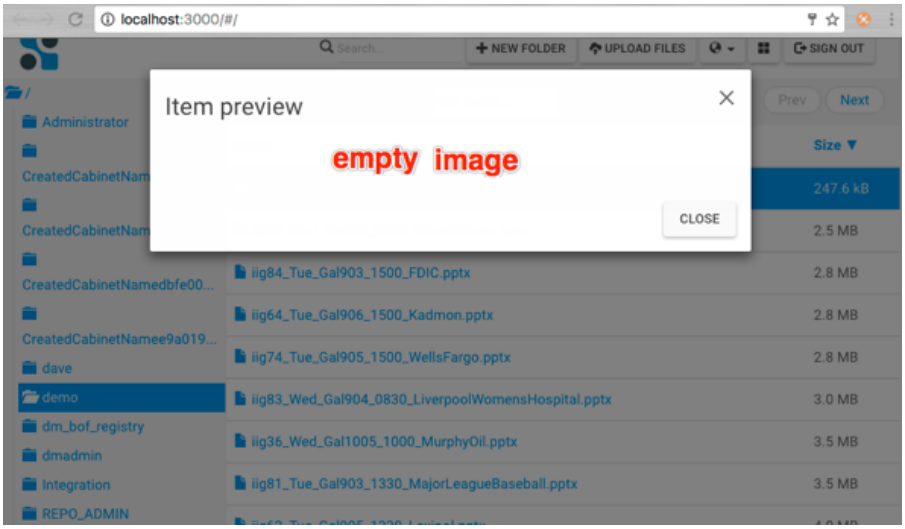
a b c

Code explanation

a. Primary Content Resource's URL can be obtained from Document Resource's link relation '**http://identifiers.emc.com/linkrel/primary-content**'.

b. It is preferred to get ACS content URL.
c. Performs a GET method.

Now if you save your work, refresh the page, and view the image document again. You will find the viewer page is promoted, but the content is still not shown.



(4) Open source code **<root>/ /src/app/filemanager/controllers/filemanager.controller.js**.

Your action at line 145 – find the link relation name for ACS content media URL

```
// TODO for MMTM R4: Find ACS content link relation from content resource START
var acsUrl = $scope.apiMiddleware.restClient.findUrlGivenLinkRelation(data, 'mmtm-
r4-find-acs-link-relation');
// TODO for MMTM R4: Find ACS content link relation from content resource END
```

a

Code hint

a. Please read Documentum REST API documentation for Content Resource.

Answer

```
// TODO for MMTM R4: Find ACS content link relation from content resource START
var acsUrl = $scope.apiMiddleware.restClient.findUrlGivenLinkRelation(data,
'enclosure');
// TODO for MMTM R4: Find ACS content link relation from content resource END
```

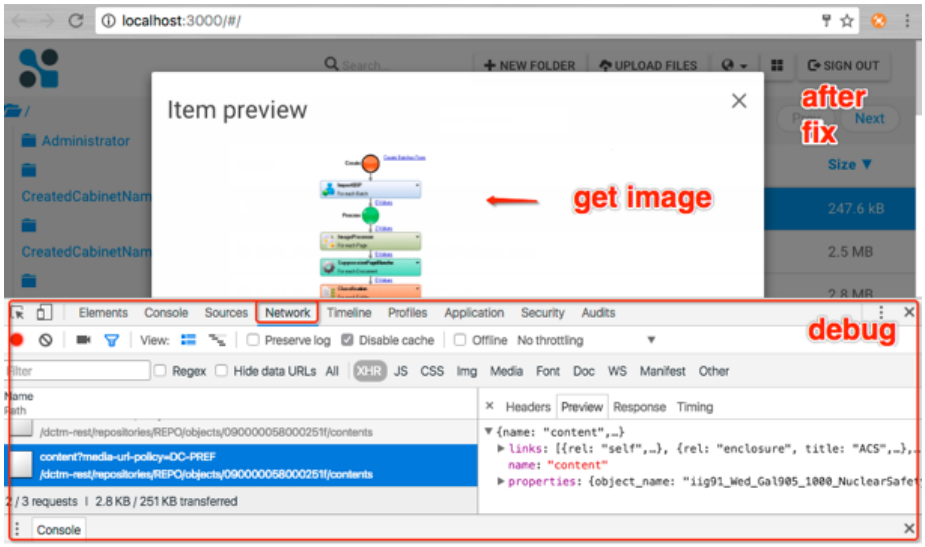
a

Code explanation

a. Primary Content Resource has link relations '**http://identifiers.emc.com/linkrel/content-media**' and '**enclosure**' pointing to the content media URL. If it's an ACS link, the link title will be **ACS**. If it's a BOCS link, the link title will be **BOCS-<Location>**. If it's a local link, the link title will be **LOCAL**.

With all above work, we can view the image in the inline viewer now. Save your work, refresh page. On an image document, right click and View Item.

.



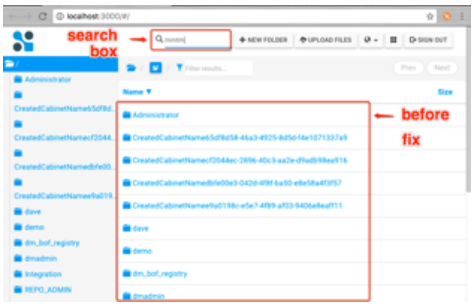
Summary

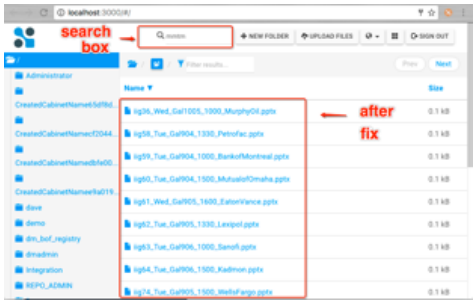
- In Documentum REST Services, a link relation is defined to discover primary Content Resource.
  - document --> content: http://identifiers.emc.com/linkrel/primary-content
- Another link relations is defined to discover Content Media Resource.
  - content --> content-media: enclosure or http://identifiers.emc.com/linkrel/-content-media

Round 5: Perform full-text search

Overview

The next round for code practice is to enable full-text search for your repository or a specific folder. You can see from the UI page's context bar, there is an input field for **Search** but it does not currently work. Here are the diagrams showing the differences before and after the fix.





Get the code

1. Open a Command window
2. Browse to **<root>**
3. Run: **git checkout –b mmtm-r5 origin/mmtm-r5 –f**
   (note: this step will override any changes you have already made locally)
4. Get back to SVC

Practice

The task for you is to complete the search function in HTML and REST client JavaScript.

(1) Open source code **<root>/src/app/filemanager/templates/navbar.html**.

Your action at line 15 – find the search function from /src/app/filemanager/controllers/filemanager.controller.js, and replace the token mmtm-r5-to-search() in below code snippet of navbar.html

```
<!-- TODO for MMTM R5: replace the search function 'mmtm-r5-to-search' START -->
<i class="glyphicon glyphicon-search"></i>

<input type="text" class="form-control input-sm" placeholder="{{'search' |
translate}}..." ng-model="$parent.search" ng-keypress="($event.which === 13)?mmtm-r5-
to-search():0">

<span/>

<!-- TODO for MMTM R5 : replace the search function 'mmtm-r5-to-search' END -->
```

a

Code hint

a. The angular directive 'ng-keypress' lets you to perform a search function after inputting the search term and pressing an **enter** key

Answer

```
<i class="glyphicon glyphicon-search"></i>

<input type="text" class="form-control input-sm" placeholder="{{'search' |
translate}}..." ng-model="$parent.search" ng-keypress="($event.which ===
13)?ftSearch():0">

<span/>
```

(2) Open source code **<root>/src/app/filemanager/services/dctm.restclient.js**. Given **Repository Resource**, please implement the search API call.

Your action at line 82 – implement the search rest api call

```
// TODO for MMTM R5: implement search rest api call START
dctmRestClient.prototype.ftSearch = function(repository, terms, path, pageNumber,
itemsPerPage) {
    return 'mmtm-r5-search-rest-api-call';  a
};
// TODO for MMTM R5: implement search rest api call START
```

Code hint

a. Please read Documentum REST API documentation for Repository Resource and Search Resource.
   - Search Resource can be obtained from Repository Resource's link relation '**http://identifiers.emc.com/linkrel/search**'
   - Search Resource supports HTTP methods **GET and POST**
   - Search Resource supports a number of query parameters, including **q**, **page**, **items-per-page**, **inline**, **locations**, and so on.

Answer

```
dctmRestClient.prototype.ftSearch = function(repository, terms, path, pageNumber,
itemsPerPage) {
    var requestUrl = this.findUrlGivenLinkRelation(repository,  a
'http://identifiers.emc.com/linkrel/search');
    requestUrl = buildUriWithQuery(requestUrl, ['q', terms, 'page', pageNumber,  b
'items-per-page', itemsPerPage, 'inline', true]);
    if (path != null && path.length > 1) {
        requestUrl = buildUriWithQuery(requestUrl, ['locations', path]);  c
    }
    return get(this, requestUrl, 'error_executing_search');  d
};
```

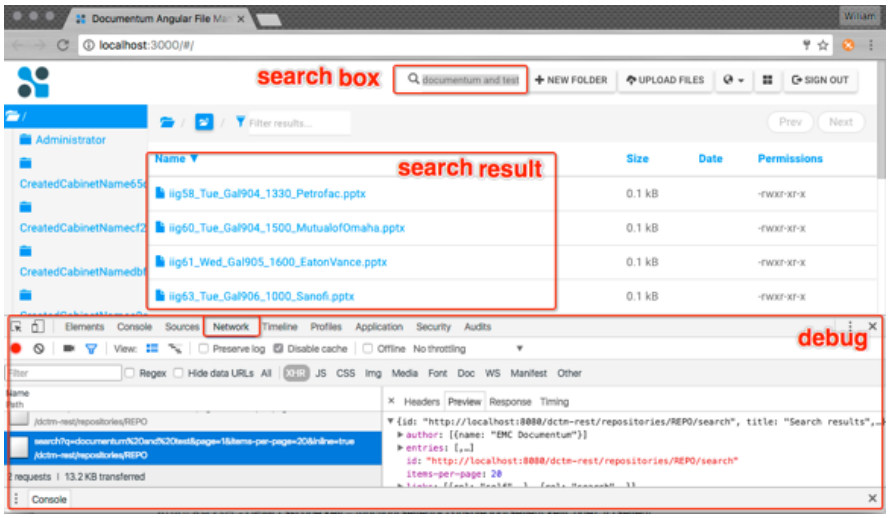
Code explanation

a. Search Resource URL can be found from **repository** resource's link relation '**http://identifiers.emc.com/linkrel/search**'.
b. Parameter **q** is mandatory; Parameters **page** and **items-per-page** determine the result pagination; Parameter **inline**,embeds the full representation of result items within the feed.
c. Parameter **locations** allows to search on a specific folder location.
d. The simple search uses HTTP method **GET**.

> ***Tip***: *Documentum REST Services 7.3 adds another method POST for the search resource, by which you can perform structure full-text search with a AQL representation. For instance, you can combine property expressions and full-text expressions together, and perform multiple facts and hierarchical facets.*

Now save your work, open the web browser, and refresh the page **http://localhost:3000**, you can now search documents by terms.

- You can use simple search language like **'emc' and ('documentum' or 'test')**
- You can select a folder first, then perform the search



Summary

- In Documentum REST Services, a link relation is defined to discover the full-text search resource.

- repository --> search: http://identifiers.emc.com/linkrel/search
- We can execute simple search language on Documentum REST Services.
- Full-text search can be based on locations, too.

Round 6: Review the complete project

Overview

All your tasks for this Hackathon lab are done. Now let's get the complete project and learn more functions like copy, move, rename, etc.

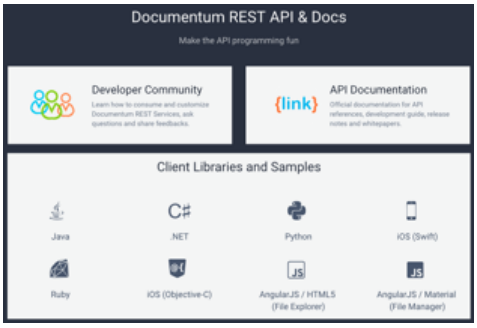
Get the code

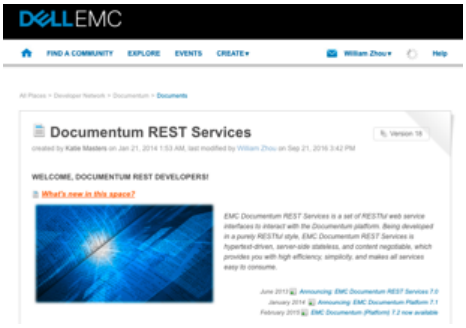
1. Open a Command window
2. Browse to **<root>**
3. Run: **git checkout –b mmtm-last origin/mmtm-last –f**
   (note: this step will override any changes you have already made locally)

Get back to SVC Summary

This sample project and its tutorials is available on GitHub: https://github.com/Enterprise-Content-Management/documentum-rest-client-angular-filemanager

To learn more about Documentum REST Services, please reach out to Documentum REST at GitHub and Documentum REST at EMC Develop Network.





Please also stay tuned for **Documentum REST Bedrock release**, which brings you more exciting features.

Cheatsheet

```
###############
##     get code    ##
###############

# get code branch for mmtm-r1
# open a new command window (1)
> cd C:\momentum
> git clone https://github.com/Enterprise-Content-Management/documentum-rest-client-angular-filemanager.git -b
mmtm-r1
> cd documentum-rest-client-angular-filemanager

# get code branch for <round x> for the 1st time
# reuse command window (1), or
> cd C:\momentum\documentum-rest-client-angular-filemanager
> git checkout -b <round x> origin/<round x> -f

# switch to existing local code branch <round y>
# reuse command window (1), or
> cd C:\momentum\documentum-rest-client-angular-filemanager
> git checkout <round y>


########################
##     build file-manager      ##
########################

# build & run file-manager
# open a new command window (3)
> cd C:\momentum\documentum-rest-client-angular-filemanager
> npm install --save-dev
> bower install
> gulp
> gulp serve


########################
##     open file-manager      ##
########################

# open file-manager in web browser
> open http://localhost:3000


#########################
##     home document URL       ##
#########################

# home document (const) path segment: '/services'
> open http://localhost:8083/dctm-rest/services
```

```
##############################################
##     part of link relations for documentum rest services      ##
##############################################

# find another resource link from the representation's link relation
# source resource --      – target resource –          --  link relation --
home doc                  --> repositories          http://identifiers.emc.com/linkrel/repositories
repository                --> cabinets              http://identifiers.emc.com/linkrel/cabinets
repository                --> search                http://identifiers.emc.com/linkrel/search
repository                --> dql                   http://identifiers.emc.com/linkrel/dql

folder                    --> child folders         http://identifiers.emc.com/linkrel/folders
folder                    --> child objects         http://identifiers.emc.com/linkrel/objects
folder                    --> child docs            http://identifiers.emc.com/linkrel/documents

document                  --> contents              contents
document                  --> primary content       http://identifiers.emc.com/linkrel/primary-content
document                  --> parent links          http://identifiers.emc.com/linkrel/parent-links

object                    --> object                self
object                    --> object (edit)         edit
object                    --> object (delete)       http://identifiers.emc.com/linkrel/delete

content                   --> content media         http://identifiers.emc.com/linkrel/content-media


##########################################
##     json media type for documentum rest services      ##
##########################################

application/vnd.emc.documentum+json


########################################
##     http methods for documentum rest services     ##
########################################

GET       # fetch
POST      # create or partial update
PUT       # complete update
DELETE    # remove


################################################
##     typical json representation for documentum rest services    ##
################################################

{
   "properties": {
      "object_name": "readme.txt",
      "title": "Read me file for beginners"
   },
   "links": [
      {
```

```
            "href": "self",
            "rel": "http://localhost:8083/dctm-rest/repositories/repo1/documents/090000123456789a"
        },
        {
            "href": "contents",
            "rel": "http://localhost:8083/dctm-rest/repositories/repo1/objects/090000123456789a/contents"
        }
    ]
}
```