**SNIA Developer Guide**

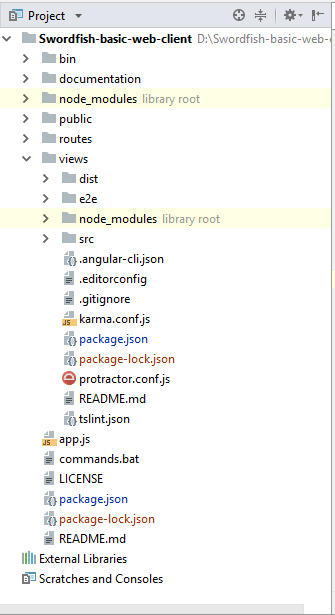
|  |  |  |
| --- | --- | --- |
| **Author** | **Date** | **Version** |
| **Sravanthi kalluri** | **26/04/2018** | **0.1** |
|  |  |  |

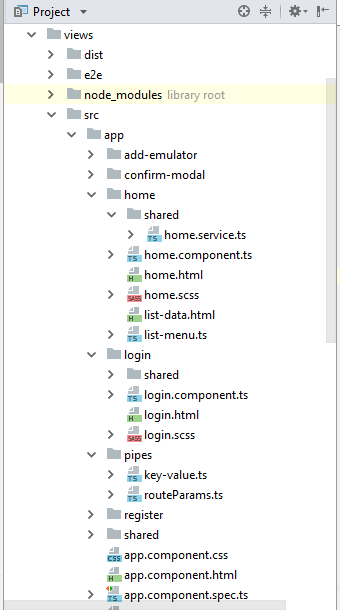
Contents

[1. Project Architecture 2](#_Toc512497680)

# Project Structure

1. The Application is build using node, express and angular-cli
2. **Package.json:** this file contains list of all the dependencies that are required to run and angular-cli
3. Run ‘npm install’ to install all the specified dependencies into local project
4. **node\_modules** will be created which contains the copy of the libraries/dependencies.





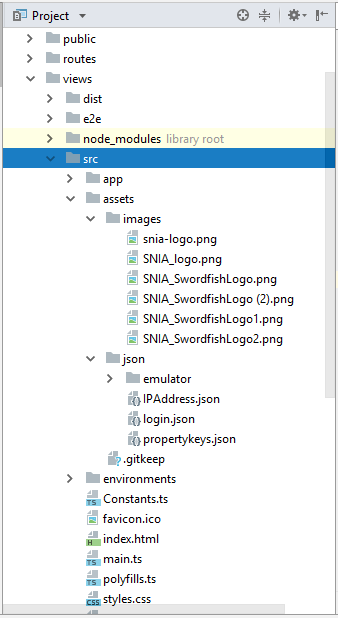


Fig: Project Architeture of basic web client

1. **views** folder contains the modules and components that are used to build the basic web client
2. **views/src** – This folder contains all the modules and components that aer used to build basic web client
3. **views/src/app –** This folder contains all the building blocks of basic web client application
4. **views/src/assets** – This folder contains all the static files,json and images that are used across the application
5. **.angular-cli.json** - contains the configurations that are required to load the angular-cli
6. **views/src/environments -** let you specify settings to customize your application behaviour

* You can define your own environments in the .angular-cli.json file.
* The default ones are:

1. **source**: use settings defined in environments/environment.ts
2. **dev**: use settings defined in environments/environment.ts
3. **prod**: use settings define in environments/environment.prod.ts

.

# 2. Data Flow

The below diagram illustrates the data flow between components

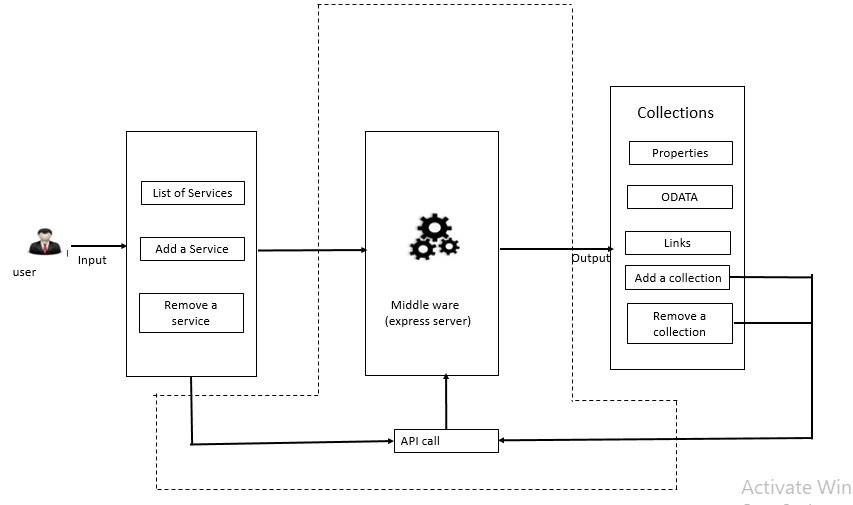


Fig: Data flow of SNIA basic web client

## Views folder

1. Any Modifications/addition of new files that relates to functional change or enhancement of the basic web client are to be done to files in this folder
2. Middle Ware(app.js): This file is used as middle ware between basic web client and the backend server
3. **Views/src/app/home** : This folder contains the logic that is used to consume the input and produce the output

# 3.Components Used

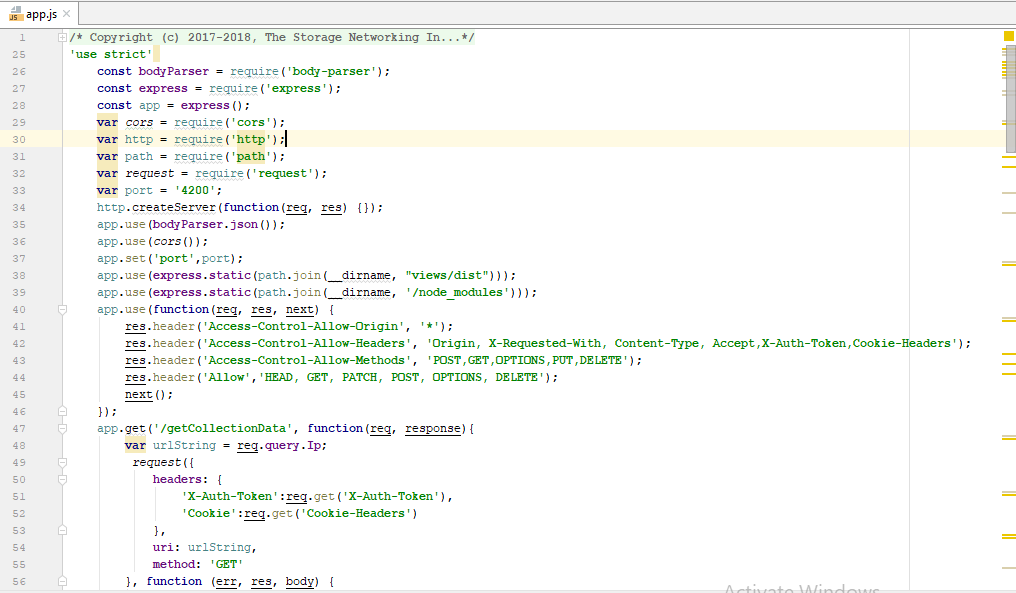
1. **app.js**

* This acts as a middle ware between angular components(basic web client ) and server(backend)
* All the required libraries are injected in to the app.js file using **require** method
* Example:

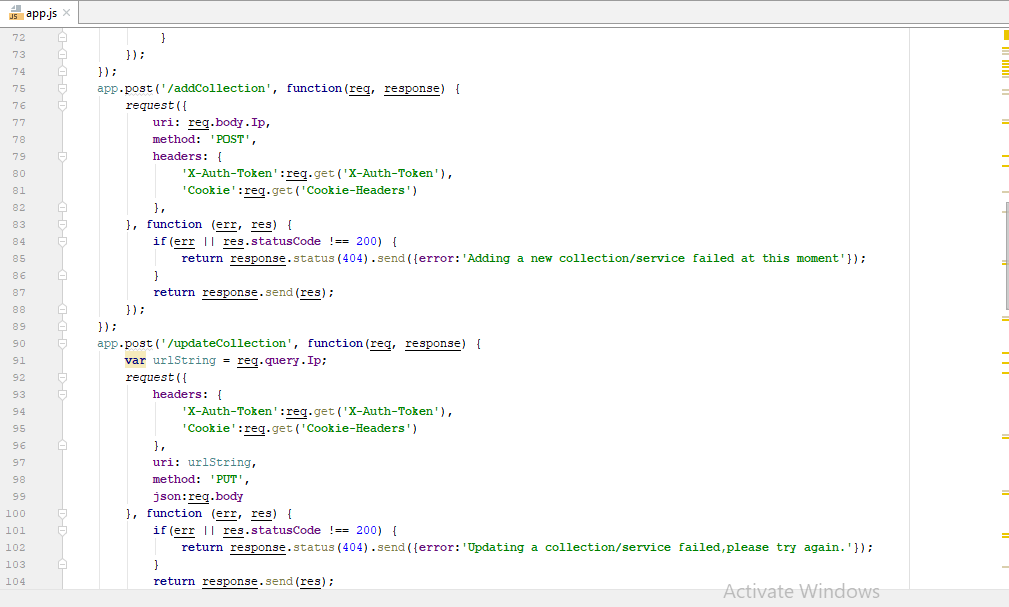
**var** *cors* = require(**'cors'**);  
 **var** http = require(**'http'**);  
 **var** path = require(**'path'**);  
 **var** *request* = require(**'request'**);

* To handle CORS issue, added required headers using app.use()

app.use(**function**(req, res, next) {  
 res.header(**'Access-Control-Allow-Origin'**, **'\*'**);  
 res.header(**'Access-Control-Allow-Headers'**, **'Origin, X-Requested-With, Content-Type, Accept,X-Auth-Token,Cookie-Headers'**);  
 res.header(**'Access-Control-Allow-Methods'**, **'POST,GET,OPTIONS,PUT,DELETE'**);  
 res.header(**'Allow'**,**'HEAD, GET, PATCH, POST, OPTIONS, DELETE'**);  
 next();  
});



* All methods that are used to communicate the view (basic web client) and backend(server) resides in this file
* CORS dependency is used to handle all the domain related issues and to transfer data from one domain to another
* **app.get ():** This method is used to fetch data from the endpoint.
* **app.post ():** This method is used to add data to the endpoint
* headers can be added and body is sent along with request



* **app.delete():** This method is used to perform any delete operations



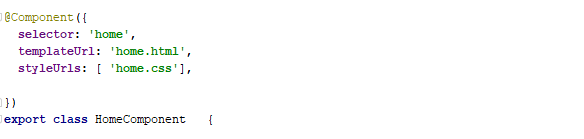
1. **app.module.ts** : This file organizes and separates the code

* imports: You can import any other modules into the imports section.
* declarations: You declare any components in your declarations. Any components used in the routing of that module, must be declared in that module. If components are used in another module, then you only list them in that other module.
* Pipes created in the application need to declare in the declarations section
* Providers: you can inject all the services into providers section
* Bootstrap: you can specify which component should bootstrap/load first into the browser in the bootstrap section



1. **home.component.ts**:

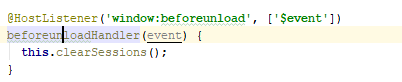
Component: It holds the configuration properties of the component like selector, names of properties, names of events and list of the injectables into the component. The value of the selector can be the same as the string to be used on HTML markup, it doesn’t need to be camel-cased



* To create a component (collection blade) dynamically ,a reference to the parent div is created with the help of @ViewChild interface



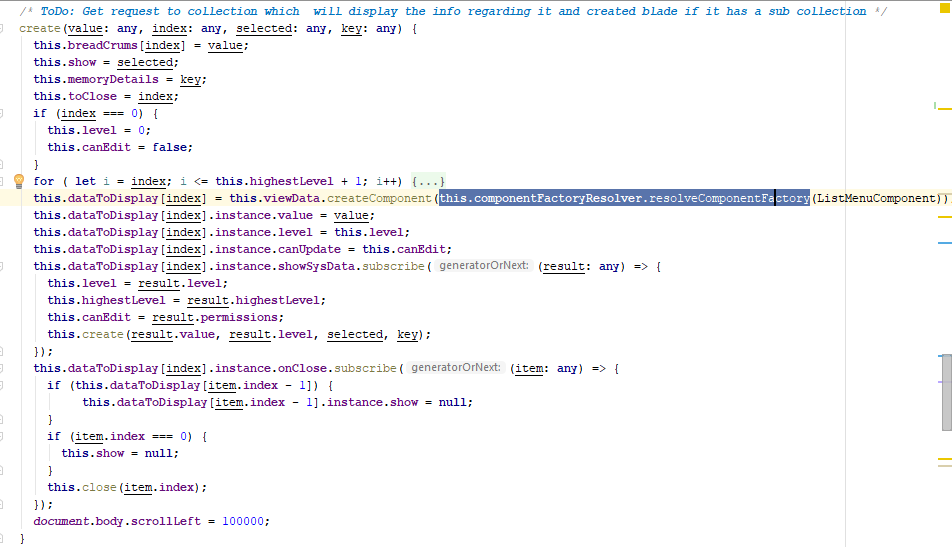
* To handle sessions and to maintain track of cookie id received while adding a service @HostListener and window events are used



* A constructor() is used to load data for html pages when component is instantiated and to inject dependencies of services

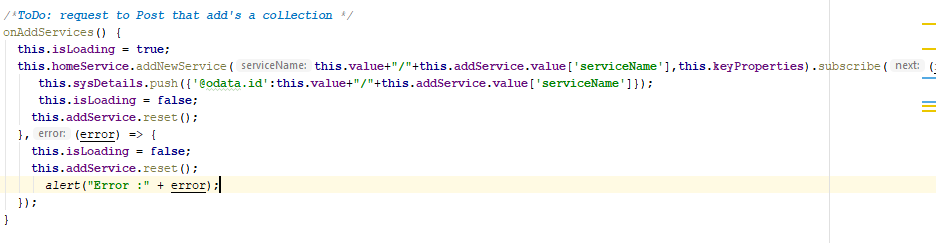


* The logic that identify the links present ,properties that exist for a collection is implemented in the create() method.
* componentFactoryResolver.resolveComponentFactory() is used to create a new blade(collection blade) if the links do exist for that collection

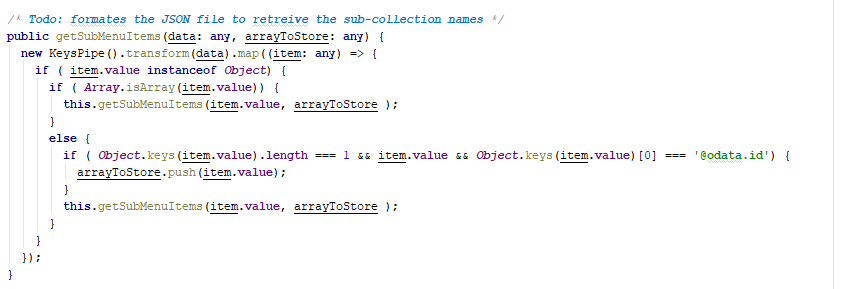


1. **list.component.ts :** The segregation of collection information runs in this component

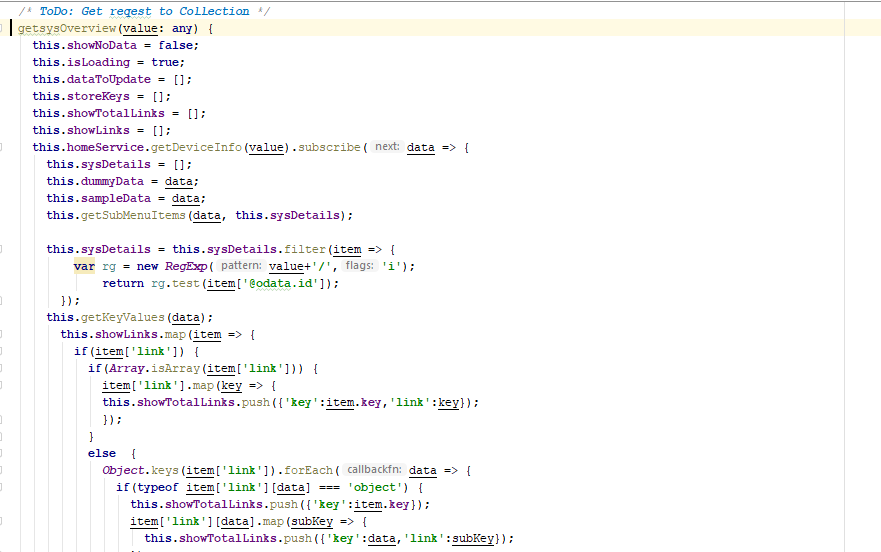
* Whenever a new service is added to the collection the response from the observable is fetched using subscribe and displays in view



* The navigable properties of collections are extracted from the @Odata string using the below method

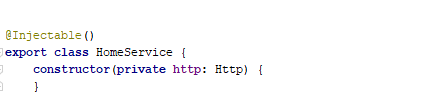


* All the properties of the collection is fetched and read entity either object/Array using getSysOverview()

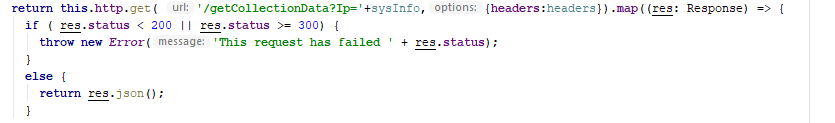


1. **home.service.ts :** The service layer from the client side is implemented in home.service.ts file

* An instance of http is created to make http calls across the browser



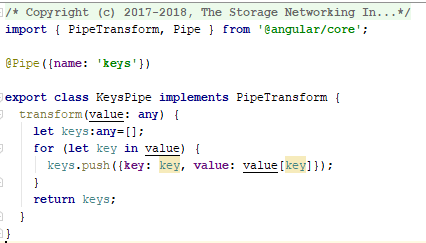
* http.get is used to make a request to fetch data from the server



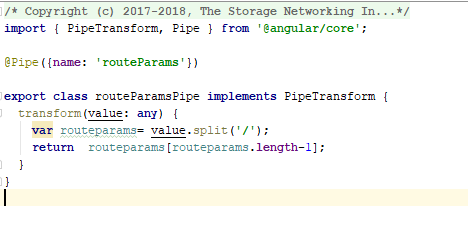
* mock API’s are built using express to avoid CORS. So, any request made from the [home.service.ts](http://home.service.ts) file will first hit the app.js file
* from app.js file the request are forwarded to the server

1. **pipes/key-value.ts** : This file is used to process and format the desired data

* The object/array entities are dived into key-value pairs where the property is stored into key and the value of the property is stored into value



1. **pipes/routeParams.ts** : To get the name of the current active collection from the link this pipe is used



1. **dist folder:** This folder contains thecompiled code of the angular application

\*\*\* build the angular application after each change/new pull request using the command cd views && ng build