VOYA Insurance - A Complete UI Development Guide

By using Aurelia JavaScript Framework

Version 1.0 (Draft)

Contents

[Why Aurelia Javascript framework for VOYA 3](#_Toc515266009)

[Aurelia framework setup 4](#_Toc515266010)

[Prerequisites 4](#_Toc515266011)

[Install Aurelia CLI 4](#_Toc515266012)

[Create a new project 5](#_Toc515266013)

[To execute the application – development phase: 6](#_Toc515266014)

[View the output in browser : 6](#_Toc515266015)

[Project folder structure 7](#_Toc515266016)

[First level folders and files details are below: 8](#_Toc515266017)

[Aurelia-project 8](#_Toc515266018)

[Src (sub folders may be changed according to the voya functional modules 9](#_Toc515266019)

[Aurelia specifications 10](#_Toc515266020)

[Development specifications 10](#_Toc515266021)

[Aurelia framework architecture: 11](#_Toc515266022)

[Development approach (Client-Server architecture): 12](#_Toc515266023)

[Local state management: Aurelia redux: 13](#_Toc515266024)

[Aurelia client side authentication : 13](#_Toc515266025)

[Development guideline: 14](#_Toc515266026)

[References 15](#_Toc515266027)

[Books 15](#_Toc515266028)

[Videos 15](#_Toc515266029)

# Why Aurelia Javascript framework for VOYA

* Create next generation JavaScript apps today
* Integrates with Web Components. No external dependencies
* Enables powerful two-way binding to any object
* Advanced client-side router with its dynamic route patterns
* Support ES5, ES 2015, ES 2016 and TypeScript
* Aurelia's extensible HTML compiler lets create custom HTML elements, add custom attributes to existing elements
* MV\* with Conventions pattern
* Easley unit testable by using karma & Vanilla
* Blazing rendering speed and great memory efficiency
* It is a programming models that are easy to learn and remember
* Integrates well with other frameworks and libraries
* It is an open source with having grate community. Commercial support options available

# Aurelia framework setup

### Prerequisites

* + Install NodeJS 4 or above

### Install Aurelia CLI

* + execute the command in the working folder “npm install aurelia-cli –g “
  + To test the aurelia installation , run the below command



* + If the installation is complete, then t he above command will shows



### Create a new project

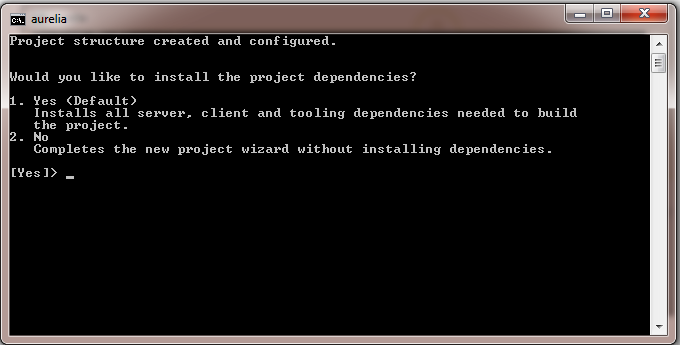
* Go to the folder where we created aurelia-cli
* Execute the command “au new yourprojectname” as below



* Enter “1” to create a basic project in ESNext (ES6) script



Enter “1” to create project – a confirmation screen



Enter 1 to install all the project dependencies. It will take few minutes to complete. This will install all the dependent node modules

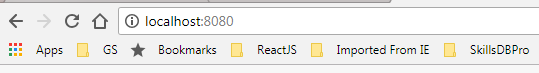
### To execute the application – development phase:

To bundle all the custom components to the app.bundle and vendor.bundle files by using the webpack module, we need to execute the below command in the command prompt working folder.



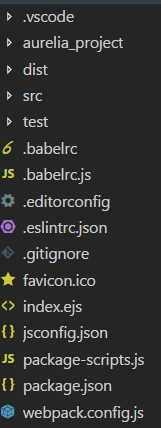
### View the output in browser :

Once the above command executed successfully ( we will be seeing bundle details in the command prompt window), we can see the output in browser window by using the localhost below url



# Project folder structure

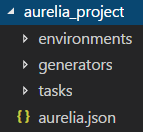
Here is the project folder structure which will be followed for the actual development



### First level folders and files details are below:

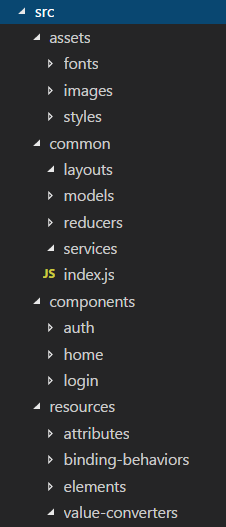
|  |  |
| --- | --- |
| Folder Name | Purpose/description |
| NEWPROJECT | Project folder name – We need to ask voyo team for the project name |
| .vscode | Visual studio code editor configuration files will be resides in this folder. Since it is an auto generated folder and files, nothing to update |
| Aurelia\_project | contains the environment configuration, generators available and tasks |
| dist | This folder contain application bundle(all the development components will be bundled in a single minified javascript file) and vendor javascript bundle and the main html page. We will copy the entire folder to the web server for the deployment . initially this folder will be empty, we will be creating bundles and transformed html page by using the command “au build” |
| src | The complete source code for the project – we will go more in depth on this folder later. |
| Test | For unit testing related files and karma configurations will be in this folder |
| Index. |  |

### Aurelia-project



|  |  |
| --- | --- |
| Environments | Will have dev, prod and stage configuration for the testing and debugging enable/disable options |
| Generators | This folder contain all the configuration and the javascripts to generate custom attribute, elements, task and value-converter |
| Task | We can create our own task (command) to execute the application . Also contain few build-in command like build, test, …. |

### Src (sub folders may be changed according to the voya functional modules



|  |  |
| --- | --- |
| Assets | Styles(all css files and sass files ), images(for all the static images) and fonts |
| Common | **Layouts** (header and footer components)  **Models** (all viewmodel json files)  **Reducers** (for state management)  **Services** – for all the common services which will be used in throughout the components  Index.js – will help us to configure for global accessibility |
| Components | Will have separate folder for each module, eg: for authentication it will be in ‘auth’ folder, for home page will be in ‘home’ …. Etc |
| Resources | For custom attributes, elements, value-conversion and binding-behaviors |

# Aurelia specifications

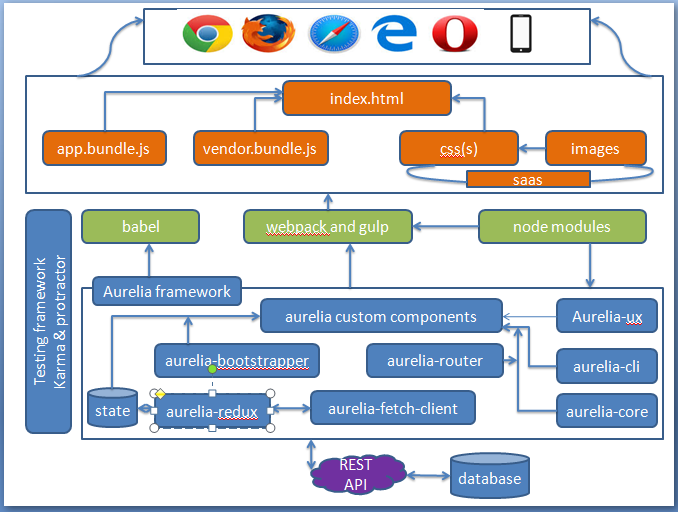
|  |  |
| --- | --- |
| UI Framework | Aurelia framework 1.2.0 |
| L icense | MIT |
| Support and maintained by | <http://bluespire.com> |
| Gold sponsors | Hogia |
| Official website | https://aurelia.io/ |
| Github | https://github.com/aurelia |

# Development specifications

|  |  |
| --- | --- |
| Development environment editor | Visual Studio Code 1.23.1 |
| UX component | Aurelia UX |
| UI unit testing framework | Karma & Protractor |
| Module bundler | Webpack 4.8.3 |
| HTTP client api | Aurelia-fetch-client |
| CSS pre-processor | SAAS |
| Scripting standard/type | ES6 |
| State management | Aurelia-redux |
| styles | Css3 |
| Validation | Aurelia-validation |
| Responsive page | aurelia-bootstrapper |
| Routing and navigations | Aurelia-router |

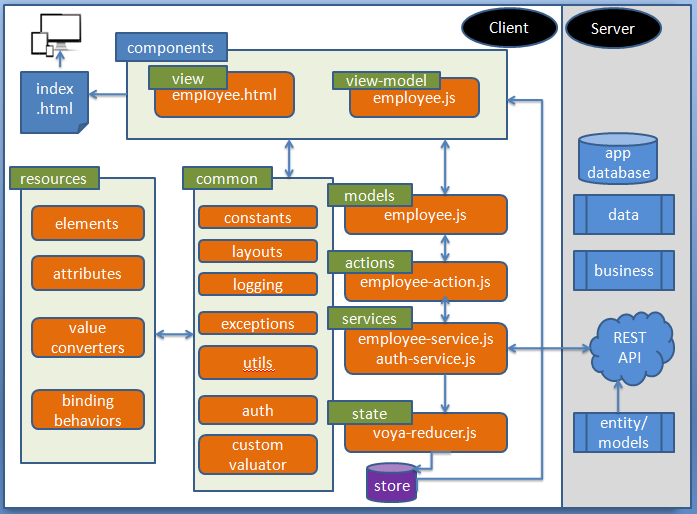
# Aurelia framework architecture:

Here below is the high level Aurelia architecture which will include webpack, ssas and Aurelia core framework libraries along with the node-modules. Since the application is having boostraper for Aurelia and ployfill libraries, there will not be any browser compatibility issues in production environment. The application can be loaded in any devices without (very minimum) much code change



# Development approach (Client-Server architecture):

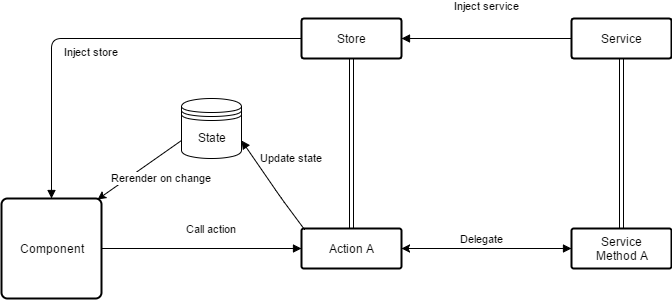
Here is the development approach and the data follow from the view(index.html) to the server Rest Api. The below design may change once we get the actual requirement from Voya



# Local state management: Aurelia redux:

We may need to use client side state management for the better client side performance and to avoid unwanted server request. The best and effienct way of implanting it in our project would be by using the Aurelia-redux library – here is the reference url for how its work in our live project

<https://www.sitepoint.com/managing-state-aurelia-with-redux/>



# Aurelia client side authentication :

On demand we will follow token based authentication for Voya project.

Ref: <https://github.com/SpoonX/aurelia-authentication>

# Development guideline:

1 Organize by feature :

his directory structure has been organized by logical features. When a feature has more than two or three sub-features, you should consider moving the sub-features to separate sub-directories.

Organization by feature keeps logically related files together, e.g. views and their viewmodels. This allows your application to take advantage of Aurelia's default convention of looking for a view and viewmodel in the same directory. It also removes the need to constantly bounce around the project's directory tree to make edits to related files

2 Naming Files and Folders

In an Aurelia application, we will typically take the name of a file's main exported class or function as the name of the file. We will convert the name from InitCaps or camelCase to dash-case.

3 Business Logic Does Not Belong in ViewModels:

Aurelia helps create applications using the MVVM (Model-View-ViewModel) design pattern. The JavaScript class that is paired with an Aurelia view is a "ViewModel." A View Model has properties and functions that can be utilized in the view via Aurelia's Databinding system. A key feature of the MVVM pattern is that business logic does not belong in the viewmodel.

4 Placing routes in their own file

Placing routes in a file of their own helps cleanly separate routing configuration. Developers will not have to hunt for route configuration when adding a new route.

5 use resources directory for global resources

The resources directory is where any global resources for your applications should be placed. The index.js file allows this directory to act as an Aurelia feature that is installed in main.js

Ref: <https://github.com/aurelia/framework/blob/master/doc/article/drafts/Aurelia%20Project%20Structure.md>

# References

## Books

There are a number of books available to help you get started learning Modern JavaScript and Aurelia:

* [Aurelia For Real World Web Applications](https://leanpub.com/aurelia-for-real-world-applications) by Dwayne Charrington
* [Aurelia in Action](https://www.manning.com/books/aurelia-in-action) by Sean Hunter
* [Learning Aurelia](https://www.packtpub.com/web-development/learning-aurelia) by Manuel Guilbault
* [Practical App Development with Aurelia](https://www.apress.com/us/book/9781484234013) by Matthew Duffield
* [Aurelia Succinctly](https://www.syncfusion.com/resources/techportal/details/ebooks/aurelia_succinctly) by Matthew Duffield
* [Practical Aurelia](https://leanpub.com/practical-aurelia) by Behzad Abbasi

## Videos

If you are looking for online video courses to help you get started learning Modern JavaScript and Aurelia, there are several great options to choose from:

* [Introduction to Aurelia Framework](https://www.udemy.com/introduction-to-aurelia-framework/) by Jon Bonso via Udemy
* [Modern JavaScript Development](https://vimeo.com/ondemand/modernjavascript) by Rob Eisenberg via Vimeo on Demand
* [Beginning Aurelia Development](https://vimeo.com/ondemand/beginningaurelia) by Rob Eisenberg via Vimeo on Demand
* [Intermediate Aurelia Development](https://vimeo.com/ondemand/intermediateaurelia) by Rob Eisenberg via Vimeo on Demand
* [Aurelia Web Development - Part 1](https://www.packtpub.com/web-development/aurelia-web-development-part-1-video) by Raja Mani via Packt
* [Aurelia Fundamentals](https://www.pluralsight.com/courses/aurelia-fundamentals) by Bryan Noyes via Pluralsight
* [Building Applications with Aurelia](https://app.pluralsight.com/library/courses/building-applications-aurelia/table-of-contents) by K. Scott Allen via Pluralsight