Developer Manual

SWEight Group - Project Colletta

SWEightGroup@gmail.com

Informazioni sul documento

Version	1.0.0
Approver	Damien Ciagola
Writers	Francesco Corti Sebastiano Caccaro Alberto Bacco Enrico Muraro Damien Ciagola
Verifiers	Francesco Corti Sebastiano Caccaro
Use	External
Distribution	MIVOQ Prof. Vardanega Tullio Prof. Cardin Riccardo SWEight Group

Description

The document contains the technical details a developer needs to comprehend and expand the software product

Change log

Version	Date	Description	Author	Role
0.1.0	2018-04-01	Backbone translated in English	Sebastiano Caccaro	*Writer*
0.0.1	2018-03-18	Document backbone	Enrico Muraro	*Writer*

Contents

List of Figures

List of Tables

1 Introduction

1.1 Document goal

1.2 Product goal

1.3 Glossary

1.4 References

1.4.1 Normative references

1.4.2 Informative references

2 Development Requirements

2.1 System requirements

2.1.1 Windows

- **CPU**: Intel X86 family;
- **RAM**: at least 2GB of RAM;
- **Disk's space**: at least 1GB;
- **Operating system**: Windows 7 or superior, 32-bit or 64-bit versions;
- **Java**: Java SE Development Kit 8;
- **Node.Js**: Node.js 10.15.1;

2.1.2 Ubuntu

- **CPU**: Intel X86 family;
- **RAM**: at least 2GB of RAM;
- **Disk's space**: at least 1GB;
- **Java**: Openjdk 8;
- **Node.Js**: Node.js 10.15.1;

2.1.3 MacOS

- **Mac Model**: all the models sold from 2011 onwards;
- **RAM**: at least 2GB of RAM;
- **Disk's space**: at least 1GB;
- **Operating system**: OS X 10.10 Yosemite.
- **Java**: Openjdk 8;
- **Node.Js**: Node.js 10.15.1;

2.2 Configuration

2.3 Execution

3 Workspace Configuration

4 FrontEnd

This section is intended to make the developer understand the working of the Colletta frontend, and to allow him or her to add functionalities to the software package. In order fully understand the contents below, the developer must have a certain degree of familiarity with React and Redux. If that's not the case, we strongly recommend the reader to at least acquire some basic knowledge on the topics.

4.1 Directory tree

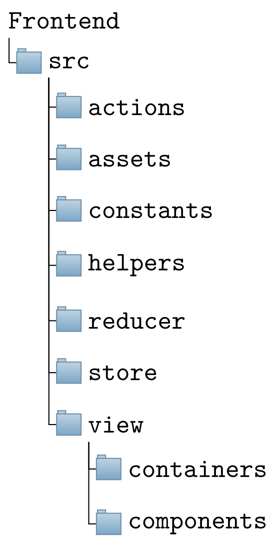

Figure 1: Frontend directory tree

Each folder contains a specific set of files:

- **actions:** the modules in this folder are responsible for creating and dispatching the actions to the reducers;
- **assets:** static files like font and images;
- **constants:** data collections and constants used in various part of the code, i.e. the label used for the translation;
- **helpers:** standard js functions or classes which have some use in the code, i.e. the label translator;
- **reducers:** all the reducers responsible for the creation of a new state;
- **store:** a single file creating and giving access to the centralized state;
- **view:** classes rendering the information in the store. They are divided in *components* and *containers*. The key point to bare in mind when talking about components and containers is the following: containers are "smart", they observe the store and can call actions; components, on the other hand, are basically just static functions.

4.2 Modify or add features

4.2.1 Components

Components extend the React `component` abstract class and implement the `render()` method. They can be viewed as a pure functions of the props passed by their father component or container. They do not have access to the store. When adding or modifying a component the following rules should be followed:

- Since the global state of the application is managed by Redux, do not use or create the local state of the component. Instead, rely solely on the props;

- Helper functions may be defined in the component class, but none of them should call action creators or external resources such as API calls;
- All components must be placed in the `src/component` folder;

When adding a new component, one can start from the following snippet:

```
import React, { Component } from 'react';
import _translator from '../../helpers/Translator';
class SampleComponent extends Component {

  render() {
    const { prop1,prop2,prop3 } = this.props;
    //Do stuff here
    return (
      <React.Fragment>
        {/* Stuff to render */}
      </React.Fragment>
    );
  }
}
export default SampleComponent;
```

4.2.2 Containers

4.2.3 Actions

4.2.4 Reducers

5 BackEnd

A Glossary