

STD – FRONT

ANGULAR | MATERIAL | NGRX | RXJS

Sommaire

1. Technologie
2. Cycle de Vie
3. Développement
4. Modules

Annexe – Base Technique

01

Technologie

Vue d'ensemble

Technologie

Base

- **Packet Mng.** npm (node.js)
- **Langage.** ts | html | css
- **CSS Pre-Processor.** sass (sass / scss)

Architecture

- **Web Framework.** angular
- **Web Component.** angular.material
- **State Management.** ngrx | redux
- **Reactive Programming.** rxjs
- **Interaction FRT / BCK.** http | rest | json
- **Lib Ref.** angular-enterprise (A creuser)

Cycle de Vie

- **IDE.** VSCode
- **Unit Test.** jasmine | karma ?
- **e2e Test.** protractor | cypress ?
- **Code Quality (Static) (// Linter)**
 - **Code Formatting.** prettier
 - codelyzer (deprecated)
 - tslint (deprecated) => eslint (for angular)
- **Host :** firebase

Inspiration

- **Angular.** jasonwatmore/alert
- **Ngrx.** loiane | ultimatecourses

Designed By - K

Library Status

- intégré | majeur | à creuser

Library (See « *package.json/dependencies* »)

Technologie

Angular

- @angular/common => http
- @angular/core
- @angular/forms
- @angular/material
- @angular/material-luxon-adapter
- @angular/router
- @angular/animation

Ngrx

- @ngrx/effects
- @ngrx/entity
- @ngrx/router-store
- @ngrx/store
- @ngrx/store-devtools (DEV Only)

Autre

- DateTime | TimeZone. luxon / intl (JS)
- Font | Icon. fonts.google
- Layout. angular-flex-layout
- Object Normalizer. normalizr
- Utils (Array, Math...). lodash

Ngrx Tiers Library

- Debug. ngrx-store-freeze (DEV Only)
- Form. ngrx-forms
- Logger. ngrx-store-logger
- ORM. ngrx-entity-relationship
- State Sync. ngrx-store-localstorage

Library Status

- intégré | majeur | à creuser

Technologie non utilisées

Technologie

- **UI Component**

- devextreme (~angular / material)
- ngx-bootstrap (bootstrap adapted for angular)
- bootstrap (impl ac class css)
- bulma
- syncfusion
- ionic

- **BCK Mock / API&DB**

- in-memory-web-api

- **CSS Pre-Processor**

- less
- stylus
- postcss

02

Cycle de Vie

(* de l'Application)

Vue d'ensemble

Cycle de Vie

Dev Environnement

- Node.JS (+ npm inclus)
- Firebase CLI : `npm install -g firebase-tools`

Configuration

Développement

Test

Build & Deploy

- Local "Dev Version" : `ng serve --open`
- Local "Prod Version" : (=> need serveur web local (Ex : lite-server))
- Remote (Cloud) : `ng deploy / firebase hosting:disable`

Pipeline

- **Static QA ? (Code Analysis)**
- **Package resolution & Build ?**
- **Test**
- **Déploiement**

Configuration

Cycle de Vie

Typescript

- Option compilateur
 - `"strict": true` ([voir ici](#))
- Variable « path » pour définir des alias pour les @module

Angular

- ?

Applicative

- **Permettre une configuration selon...**
 - Une instance d'env « DEV/REC/PROD »
 - Un « Site Client »
- **Localisation de la Configuration**
 - Fichier « Environnement.ts »
 - @material ->
- **Élément Configurable**
 - URL Service Business
 - URL Service Tiers (Map, GED, eSign...)
 - Fichiers (taille...), Timers (autologout...)

- **Tester c'est Doubter ;p**
- **Test UI ?**
- **Test Unitaire**
 - Test des méthodes « publiques » & « de plus d'une ligne »
 - Pas de test des librairies tierces
 - Couverture : XX%
- **Sonar**

03

Développement

Convention de Nommage

Développement

- **Dossier | Fichier**

- Minuscule + Séparateur = « - »
- Nom : <nom-fichier>.<composant>.ts (Ex:)

- **Module**

- **Alias** : Préfixé par « @ »

- **Classe | Object | Type | Variable**

- Casse : CamelCase (maVariable, monObjet)
- Observable : Suffixé par « \$ » (obs\$)
- Private : Préfixé par « _ » (_maVar)

Annotation

- **TODO** : Code à retravailler
- **TO_CONF** : Code à adapter selon le besoin

Alias

- @alert, @env, @enum, @material, @layout, @form, @loader, @router, @timer, @token
- @core, @account, @shoppingList, @product
- @action,
- @package, @log, @deploy, @style, @lint

- **Base Technique**

- Essentiellement des modules Angular

- **Type**

- Shared : modules partagés
- Core : Cœur de l'application
- Feature : Fonctionnalités

- **Import**

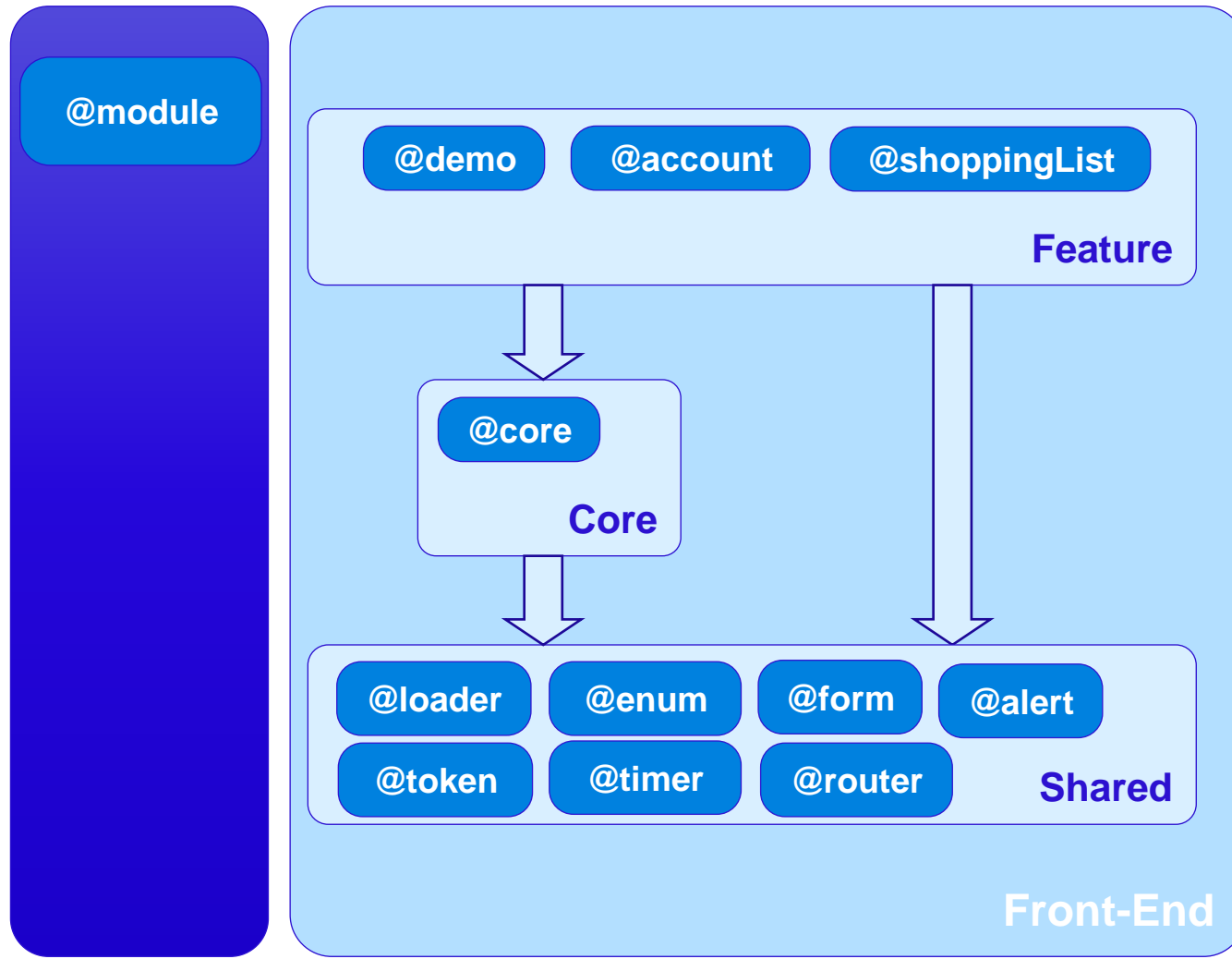
- 1 Alias : @NomModule
- Import * as from <Module> from '<alias>...'
- « index.ts » : Référence Effects, Composants
- Lazy-Loadés -> cf. doc angular

Arborescence Code

- **Component**
- **Effect**
- **Guard**
- **Model** (Contient Model & Enum)
- **Page** (/!\ != « Page » dans SPA)
- **Service** (Contient les API)
- **Store**
 - Actions
 - Reducers
 - Selectors
 - State
 - index.ts
- <Feature>-router.module.ts
- <Feature>.module.ts

Module / Dépendance

Développement



Architecture

Ngrx

Développement

Store

- Un fichier index pour référencer ce que le store fourni
- ngrx rules

State

- Un **global state** (pas de local pour l'instant)
- Data **normalisés** (cf. redux)
- Ne mettre que des plain object dans le state -> (=> pas de method, d'action...) (cf. redux)

Reducer

- Comment update le state ? (cf. redux)

Selector

- ?

Action

- **Nom.** <action><Entité>Action (ex : submitFormAction)
- **Définition.** ngrx (Version >8.x)
- Emissible par : « **Component** » / « **Api** » / « **Store** »
- Créer au plus près de l'émetteur
- Module @action fournit des helpers
- **Action prédéfinis.** @form / @button

Effect

- **Nom.** Nommer selon ce que fait l'effect
- **Règle.** manipulating effects
- Gérez l'activation de l'effect via
 - **ofType()** pour filtrer selon le type de l'action
 - **Filter(Fn)** pour le payload de l'action ou autre
 - withLatestFrom si besoin du state

Module

- **Nom.** <Xxxx>Module (Ex : FormModule)
- **Import.** Effect | Store | Component

Composant

- **Selector**
 - Dans « **Shared** » :
 - **Modèle** : <k-<nomModule>-<nomComposant>>
 - **Exemple** : <k-form-field-input>
 - Dans « **Feature** » :
 - **Modèle** : <<nomFeature>-<nomComposant>>
 - **Ex** : <demo-alert>
- **Héritage**

- **Module.** load method -> lazyloaded
- **Router.** Sync avec ngrx-router-store
- **Form.** Reactive Form | Forms (NON)
- **I18n.** Tag... (à creuser)
- **Accessibilité.** aria-label... (à creuser)
- **Resolvers.** (à creuser)

Autre

- **Service (DI)**
- **Router**
- **Guard**
- **Directive**
- **HTTP Client**
- **HTTP Interceptor**
- **Pipe**
- **Resolver**

A Creuser

- **Web Worker**
- **Service Worker**
 - Pour PWA !
- **Angular Universal**

Autres Fonctionnalités

Développement

Layout

- **Library.** angular-flex-layout
 - UI Example | otherDoc
 - Declarative API | Responsive API
- Chaque feature HomePage à la responsabilité de son layout (à noter que le root Component d'un module se trouve de facto dans un Flex/Column)
- Distinction de 2 usage (Choix Conception) :
 - Mobile : -> **.xs**
 - Le reste -> **.gt-xs**
- Implémentation
 - Soit dans un composant commun ac flex-layout

Code Quality

- **Analyse Statique**
 - Formatage du code
 - Utilisation Prettier & ESLint

Log

- **Library.** **ngrx-store-logger**
- **TODO**

Environnement Supporté

- **Browser** : Chrome, Edge, Firefox, Opera ?
- **Device** : PC, Mobile, Tablette ?
- **Ecran** : Longueur, largeur, définition, résolution ?

04

Modules

- **@action.** Standardise le nom des actions NgRx et la façon de les créer
- **@alert.** Gère les alertes pour l'utilisateur avec différents niveaux de criticité
- **@enum.**
- **@form.** Gère les formulaires (état, validation, persistance...) (Basé sur ngrx-forms)
- **@loader.** Gère l'état de chargement de l'application et l'affichage des indicateurs correspondants
- **@material.** Fournit des composants graphiques (Bouton, tableau...) (Basé sur angular.material)
- **@router.**
- **@timer.**
- **@token.**

Fonctionnalité

- **Thread.** Zone d'écran ou s'affiche des alertes
 - **Nb thread affichable** : 1
 - **Nb Alert / Thread** : 1
- **Dismissal.**
 - Manuelle : OUI -> Appui icone « X »
 - Auto : OUI -> Après 'chgt de route' | 'X sec'
 - Style (Instant / Fade) : Fade
- **Component.**
 - Style : Couleur selon Criticité
 - Criticité : « success, info, warn, error »
 - Contenu : message

Implémentation

- Material snackbar
- Couleur selon criticité : Bootstrap

Action

- `triggerAlertAction`
- `dismissAlertAction`
- `keptAfterRouteChangeAction`

@enum

Modules

Fonctionnalité

- Gère des énumérations dynamiques

Implémentation

- 1 API REST

@form / Présentation

Modules

Concept

- **Form.**
 - Est un **composant configurable**
 - Peut contenir [1..*] **Field**
 - **A un état** résultant de l'état de ses fields
 - Est **persistable & validable**
 - Emet une action à la soumission
- **Field.**
 - Est un **composant configurable**
 - **Type** : Input, Select, Date, Checkbox...
 - **Validation** :
 - Statique & Dynamique
 - Configurable
 - Message d'erreur **-> injecter module ?**
 - **Persistence** : désactivable (ex : Password)

Composant

- **Form.** <k-form>
- **Field.** <k-form-field-**<type>**>
- **GroupField.** <k-form-group-field-**<name>**> (password)
- **Button.** <k-button>

Selector

- **selectForm(formId)**
- **selectFormValue(formId)**
- **selectControl(formId, ctrlId)**
- **selectControlValue(formId, ctrlId)**

Layout

- 1 grid | 1 column
- Field
 - Min-width : **240px**
 - Width : **100%**

Action

- **resetFormAction**
- **clearFormValueAction**
- **formValidatedAction**
- **submitFormAction**
- **validateFormAction...**

Validation

- Fonction « ngrx-forms »
- HomeMade
 - **mustMatch(ctrl, ctrlRef)**

@form / Utilisation

Modules

1. Module

- Importez le module « FormModule »
- Créez un **component** (x.component.ts + .html)
- Créez un **effect** (x.effect.ts)

2. Formulaire

- Placez un `<k-form>` et spécifiez :
 - **(obligatoire)** un identifiant unique **[formId]**
 - la persistance dans le global state (`no-persist`)
 - la validation des champs (`no-validate`)
 - **Ex** : `<k-form [formId]='my_form' no-persist no-validate></k-form>`

3. Layout

- Par défaut, un `<k-form>` est une grid d'une colonne
- Vous pouvez redéfinir le layout du formulaire en ajoutant un `<div>` et en utilisant 'angular-flex-layout'

4. Contenu

- **Champ(s).** Placez [1..*] `<k-form-field-XXX>` & spécifiez :

Identification

- **(obligatoire)** un nom unique ds le formulaire **[ctrlName]**
- un **[formId]** (si != du `<k-form>`)

Persistence

- La persistante dans le global state : `[unpersist]="true"`

Usage Info

- Label
- Placeholder
- Value

Validation Rules

- Format (password | email | number)
- Required
- [1..*] Fonction de Validation

- **Bouton(s).** Placez [1..*] `<k-button>` de soumission

5. Effet (x.effect.ts)

- Ecoutez une action (Ex : `formValidatedAction`)
- Filtrez sur l'**identifiant du formulaire** (`filter()`)
- Récupérez les **valeurs du formulaires** dans l'**action** ou via un **selector**

@form / Implémentation

Modules

Module en Dépendance

- LoaderModule (Pour BasicButton)
- LuxonModule (Pour DateField)
- MaterialModule (mat-form-field, mat-button...)
- NgrxFormsModule (FormGroupState, ...)

Technique

- **Angular**
 - Component, Dependancy Injection, Directive
 - Content Projection, Component Interaction
- **Typescript** : Classe/Héritage
- **Ngrx-forms** : UI Ex | Actions | Code | Code Ex
 - * actions existantes ds « ngrx-forms » ont été redéfinies dans le module @form (la maj de l'état en résultant aussi via un reducer), Ceci parce que les actions 'ngrx-forms' sont implémentées avec la méthode de Ngrx V<8.x (Version actuelle |) alors que les actions de l'app sont basés sur la version post 8.x
 - /\ ngrxEnableFocusTracking non utilisé

Contenu

- **Composant**
 - Model : FormComponent & FieldComponent
 - Button
 - Field
 - Field-group
- **Service**
 - **error-message** : Gère les messages des erreurs
 - **validation-fns** : Gère les fonctions de validation
- **Store**

@loader

Modules

Fonctionnalité

- **Activation / Désactivation**
 - **Activation** : à l'envoi d'une requête HTTP
 - **Désactivation** :
- **Identification**

Implémentation

- **Component**
- **Store**

Utilisation

- **Quand ?**
 - Now : dès qu'il y a un appel HTTP (via interceptor http)
- **Ou ?**
 - Dans les Submit Bouton des « @Form »

Bouton & Lien

Icon / Font

- **Icon** : liste / user guide

Form

Table

- **Contenu** datasource
- **Colonne et lignes**
- **Pagination**
- **Tri**
 - S'appliquer au contenu affiché, (Si * Page ?)
 - Est sélectionnable pour une ou *colonne)
- **Filtre**
- **Sélection**

Autre

- **SideNav, Toolbar, Accordéon...**

@timer & @token

Modules

Fonctionnalité

- Déclaration d'un Timer
- Emet l'action fourni à la fin du Timer

Action

- `defineTimerAction`
- `deleteTimerAction`
- `timerDefinedAction`
- `timerDeletedAction`
- `timerEndedAction`

Fonctionnalité

- Déclaration d'un Token
- Appel l'Api de validation des Token

Action

- `validateTokenAction`
- `deleteTokenAction`
- `tokenValidatedAction`
- `tokenInvalidatedAction`

Fonctionnalité

- Critère de Recherche
- Résultat de Recherche
 - **Pagination**

Implémentation

- **Composant « Critères de Recherche »**
 - `<div class="searchbar-filter">`
 - `<div class="searchbar-filter-list">`
- **Composant**
 - **filter-shell** : enveloppe standard des filtres
 - **filter-select-tree-view** : (Instance) - arbre de checkbox
 - **filter-search-tag-box** : (Instance) - Présente un champ libre avec 'autocomplétion'

Merci !

HTML

- **Balise, Attribut** `<balise></balise>`
- `<head><body>`
- `<a>`, `<p>`, `<button>`, `<video>...`

CSS

- **Selector, attribute, class**
- **Pseudo-elt** (`:`), **Pseudo-class** (`::`)
- **Display.** inline, block, ***flex*** / ***grid***...
- **Unité.** px, em, rem, %, fr
- **Flex.** container / direction / wrap / grow...
- Position / margin / padding...

TS

- **Variable, type, enum**
- **Class, property & function**
- **Interface & Héritage**
- **Public | Privée**
- **Modifiers** (readonly, optionnal...)
- **Module**
- **Decorator** (~Annotation de classe)
- ...

Angular

- **Module**
- **Component (.html, .css, .ts)**
- **Service : Injectable**
- **Model, Enum**
- **Guard, Interceptor**

NgRx

- **Store, State**
- **Action, Reducer, Selector, Effect,**
- **Entity, Adapter**
- **Router-store, Store-devtools**

Material

- **mat-form-field**
- **mat-icon**
- **mat-button**
- **...**

Rxjs

- **Observable / Subject**
- **tap, map, filter, of, first**
- **switchMap, exhaustMap...**