Vanrise Coding Conventions

Table of Contents

[**1** **Introduction** 2](#_Toc15386814)

[**2** **General Guidelines** 3](#_Toc15386815)

[**2.1** **Naming conventions** 3](#_Toc15386816)

[**2.2** **Client Grid Tips** 3](#_Toc15386817)

[**2.3** **UI enhancements** 3](#_Toc15386818)

[**2.4** **UI Alignments** 3](#_Toc15386819)

[**2.5** **Multi-layers UI communication** 3](#_Toc15386820)

[**2.6** **Loader uses and Loading Dependences** 4](#_Toc15386821)

[**2.7** **General Rules** 5](#_Toc15386822)

[**2.8** **Project Structure (Old & new Convention)** 6](#_Toc15386823)

[**2.9** **Architecture conventions (Always start with single abstraction / use wrapper class for abstraction** 6](#_Toc15386824)

[**3** **Static Guidelines** 7](#_Toc15386825)

[**3.1** **Grid and Drill Downs** 7](#_Toc15386826)

[**3.2** **Search Pages** 7](#_Toc15386827)

[**3.3** **Selectors** 7](#_Toc15386828)

[**3.4** **Extensions** 7](#_Toc15386829)

[**4** **Dynamic Guidelines** 8](#_Toc15386830)

[**4.1** **Definition – Runtime abstraction** 8](#_Toc15386831)

[**4.2** **Main grid** 8](#_Toc15386832)

[**4.3** **Sub views** 8](#_Toc15386833)

[**4.4** **Actions** 8](#_Toc15386834)

[**4.5** **Bulk Actions** 8](#_Toc15386835)

# **Introduction**

The first step of making a framework we should think about maintainability, complexity and coding quality. To satisfy these three main concepts in programming we should follow a set of guidelines/conventions.

These guidelines were put according to company vision and the way that make high productivity in small interval of time.

In addition, all conventions were put after discussions and years of experience in applying best practices to have such great framework.

# **General Guidelines**

## **Naming conventions**

Let we have an entity names: Country

1. Management pages: CountryManagement
2. Editors: CountryEditor
3. Selector: {solution}-{module}-country-selector
4. Grid: {solution}-{module} -country-grid
5. Abstract settings: {solution}-{module}-county-setting
6. API Controller: CountryContoller
7. Business Manager: CountryManager
8. IDataManager: ICountryDataManager
9. DataManager: CountryDataManager

Note: solution will be replaced by solution symbol ex: Tone: whs , Vanrise: vr, Retail: retail.

Module will be replaced by module name example : common, genericdata .. etc

## **Client Grid Tips**

It’s preferable to put the dataItem of the grid in an object to collect the clean object.

Ex: $scope.datasource.push({Entity:dataItem});

## **UI enhancements**

1. Don’t use “vr-loader” or “vr-disable” on directive wrapper, try to use a span outside the directive wrapper.

## **UI Alignments**

We should be care about the alignment of all controls in the page or editor, they should be aligned and same size and the trick behind the alignments is the way we use vr-row and vr-columns.

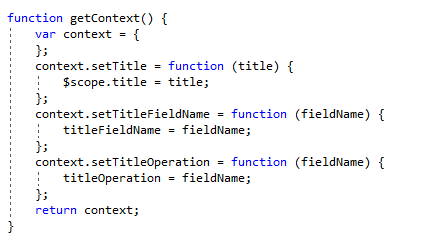
|  |  |
| --- | --- |
| Wrong behavior | Correct behavior |
|  |  |

## **Multi-layers UI communication**

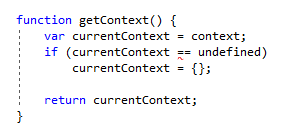
The most important thing in multi-layer UI for example we have two or three pop-ups and we need to let the third one communicate with the first one we need to create a context as an object that have only functions van be passed for all editor and each editor will call the function needed.

In addition, in each directive or editor we need to create another context that contain a copy from the previous one plus the additional functions to be passed for any child control.

1. In main editor:



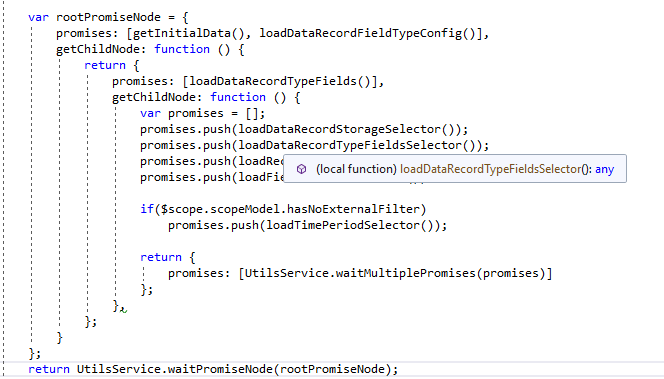
1. In sub editor



## **Loader uses and Loading Dependences**

Loader is the most important thing in the application to prevent user from taking any action before we finish loading the pages.

1. Always use PromiseNode



1. Dependent Selectors
2. Loading when using directive wrapper

## **General Rules**

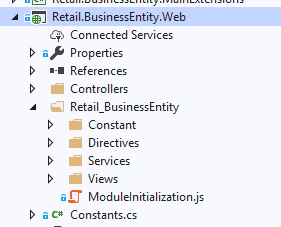
The below rules should be applied instead you have a specific permission from your supervisor.

1. Not allowed to use settimeout function.
2. Not allowed to use watchers in business directives.

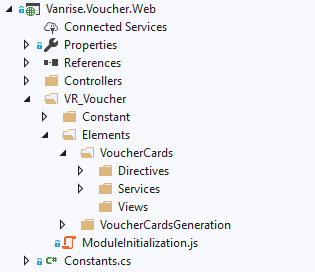
## **Project Structure (Old & new Convention)**

Rules:

1. Don’t mix between old and new.
2. Any new project created should follow the new convention.
3. Old Hierarchy:



1. New Hierarchy



## **Architecture conventions (Always start with single abstraction / use wrapper class for abstraction**

# **Static Guidelines**

In this section we will talk about the guidelines that help us to build a static page following the best practices and the latest conventions used.

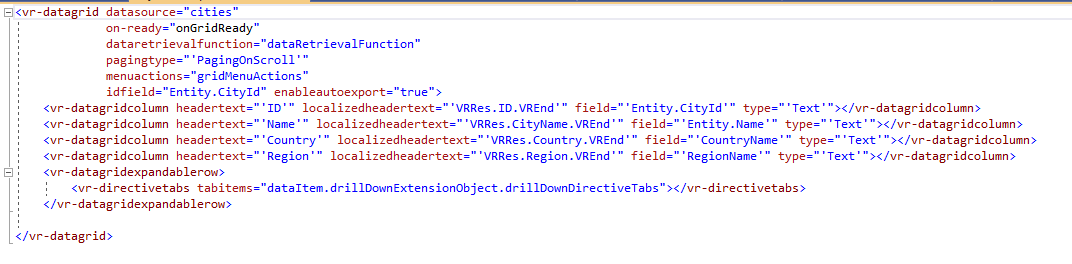
## **Grid and Drill Downs**

1. When we want to develop a static grid, the grid implemented as a directive with load method.
2. Drilldowns should be implemented in the form of registration to allow adding different drill downs from several modules.

## **Search Pages**

Any drilldown should be considered as search page so we need to build two directives one for grid and the second for search and we use the grid inside it as following:

1. Grid Directive

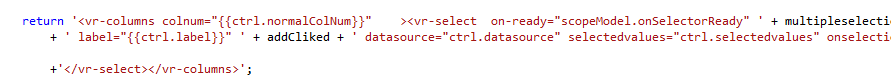


1. Search Directive



## **Selectors**

It’s required to use vr-columns in the selectors directive and pass normal-col-num attribute from outside.



## **Extensions**

# **Dynamic Guidelines**

## **Definition – Runtime abstraction**

1. When we want to use an abstract class for definition and runtime purpose, we should put the runtime directive as a property in the definition class.

## **Main grid**

1. Developing such grid required a definition entity having the settings for grid columns, these columns should be evaluated server side before loading the grid. (ref: Check Generic BE Grid).

## **Sub views**

The implementation of sub views should be in the same way as registration, but in dynamic behavior we should return the valid sub views on the data Item entity that allow us to find all sub views needed to be shown.

This allow us to hide and show any sub view not match with a specific record under certain condition.

## **Actions**

## **Bulk Actions**