Table of Contents

ISSUE-TRACKER SYSTEM GUIDE

1. INTRO

1.1. Purpose

The purpose of this guide is to provide description of the existing issue-tracker System and it's architecture

1.2. Audience

Any givien instance of the issue-tracker should have ONE and only ONE person which is responsible at the end for the funtioning of THIS instance - so think carefully before attempting to take ownership for an instance. The author(s) of the code are not responsible for the operation, bugs or whatever happens to a new instannce. As a responsible owner of an instance you could create, share and assign issues to the authors of the source code, yet there is no service level agreement, nor even promise to help.

2. ARCHITECTURE

2.1. IOCM architecture definition

The Input-Output Control Model architecture is and application architecture providing the highest possible abstraction for allmost any software artifact, by dividing its components based on their abstract responsibilities, such as Input, Output , Control and Model.

2.1.1. The Control components

The Control components control the control flow in the application. The instantiate the Models and pass them to the Readers , Converters and Writers for output.

2.1.2. The Model components

The model components model the DATA of the application - that is no configuration, nor control flow nor anything else should be contained wihing the model.
Should you encounter data, which is not modelled yet , you should expand the Model and NOT provide differrent data storage and passing techniques elsewhere in the code base ...

2.1.3. The Input Components

The Input Components are generally named as "Readers". Their responsibility is to read the application data into Model(s).

2.1.4. The Output Components

The Output Components are generally named as "Writers" Their responsibility is to write the already processed data from the Models into the output media .

2.1.5. The Converter Components

The Converters apply usually the business logic for converting the input data from the Models into the app specific data back to the Models.

2.2. Multi-instance setup

The multi-instance setup refers to the capability of any installed and setup instance of the issue-tracker application to "know" its version , environment type - developement , testing and production) and owner.

2.2.1. Multi-environment naming convention

Each database used by the issue-tracker application has an <<environment abbreviation>> suffix refering to its environment type. Running application layers against different db versions should be supported as much as possible.

2.3. Sofware architecture

2.3.1. Front-End

The Mojolicious Web Framework runs on top of a perl instance, which serves the back-end requests and passes back and forth json, as well as the ui Mojo templates dynamically, which combined with the vue template create the generic ui.

2.3.2. Back-End

The id's of the tables which ARE VISIBLE to the end users ui are big integers, which are formed by the concatenation of the

year,month,day,hour,minutes and second in which the row in the table is created.

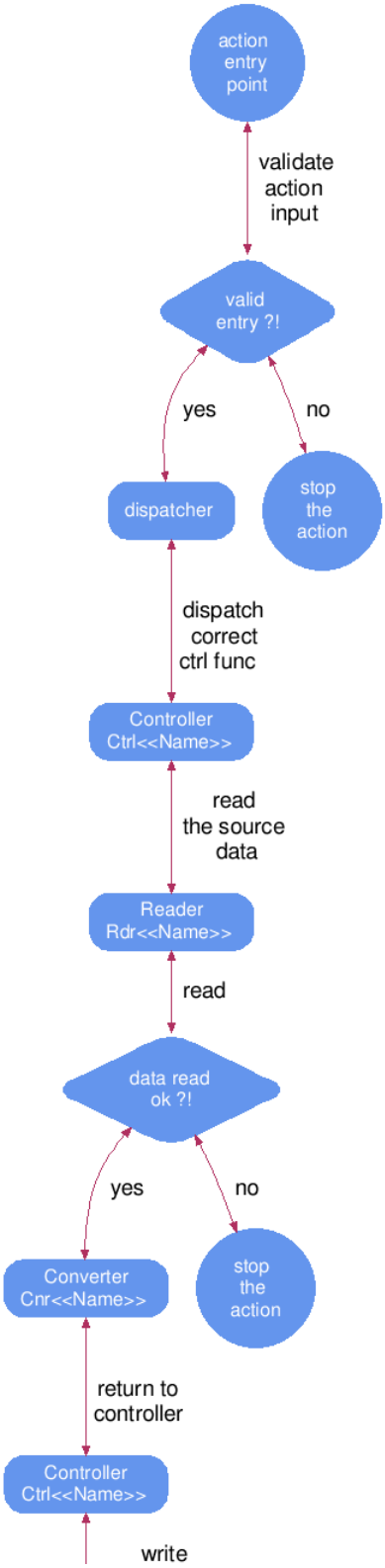
3. APPLICATION CONTROL FLOW

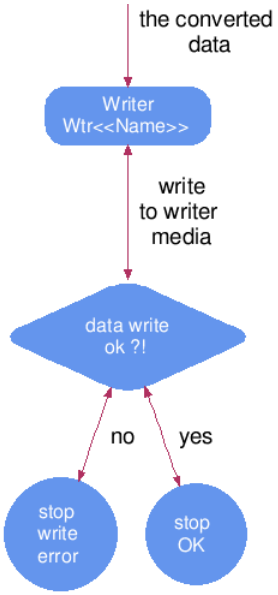
This section provides a generic control flow description for the shell based and ui based control flows.

3.1. Shell control flow

The shell control flow is based on the control model input output architecture.

Figure: 1 issue tracker control flow





control flow in shell actions' execution