Gmail Pablo Valdunciel <valdunciel98@gmail.com>

Julia Code structure

Alberto de Miguel Valdunciel <alberto.m.valdunciel@gmail.com> 1 de julio de 2022, 14:12
Para: valduncielpablo@gmail.com

# Modules
- Filename capitalised.
- Structure of the module, in order and delimited by header:
 - 3rd Party dependencies "global" to the module.
 - DOCYET dependencies "global" to the module.
 - Inclusion of files.
 - Files follow the same structure to list
 - Exported references (which are considered the module's public API).
 - Module initialization, if needed.
 - Auxiliary functions
 - Main functions. Might make the module's Public API in the case of a module.

# Sample file: SuggestionsController.jl

```
"""
 SuggestionsController

Groups functions that make use of prefix trees.
"""
module SuggestionsController
@info "Loading SuggestionsController"

# Framework Dependencies
# -----------------------
using Genie, Genie.Renderer.Json, Genie.Requests

# Base Dependencies
# -----------------------
import Base: values

# 3rd Party Dependencies
using DataStructures

# DOCYET Dependencies
# -----------------------
using Strigiformer
# Probably better to do "using" instead of "import" with so many imported functions.
import DocyetUtils: @hose, Maybe, @builder, @setter, slice, deep_symbolize_keys, take
```

```julia
# Lib Dependencies
# -----------------------
using LanguageUtils
using Tries
using Decompounding
using Ontology


# Inclusions
# -----------------------
include("models/Models.jl")

# Module Constants
# -----------------------


# Auxiliar Functions
# -----------------------
function build_trie_from_ontology_class(o::OWL.Ontology, class::Union{String,OWL.Class},
transformer::Function, T)
 @hose findall_by_class(o, class) |>
 map(transformer, _) |>
 Dict |>
 deep_symbolize_keys |>
 build_trie(_, T)
end


# Exported references
# -----------------------
# @info "Creating Suggestion Tries..."
# @time const SUGGESTION_LISTS = Dict(
# "preconditions" => build_trie_from_ontology_class(o, "docyet:Precondition", symptom_transformer,
PreconditionModel),
# "symptoms" => build_trie_from_ontology_class(o, "infermedica:Symptom", symptom_transformer,
SymptomModel)
# )


# API Functions
# ------------------------
function ICD()
 suggest_ICD(jsonpayload("code")) |> json
end

function suggest()
 @show params()
end
```

end