Java Home Project

Implementation of the rest API done on top the spring boot framework.

This allows very quick development the rest endpoints and adds the powerful DI abilities of this framework.

This document does not describe the classes provided by spring application wizard.

# UI

## UIController

This is the Rest controller which defines the two end points

@GetMapping("/names")  
public ResponseEntity<String> getNames(@RequestParam("prefix")String prefix,   
@GetMapping("/reset")  
public ResponseEntity<String> reset()

These method will delegate the calls to BE server using the Rest Client object

## BERsClient

This class is on duty to delegate calls to BE server

There are two methods

Single<ResponseEntity<String>> getNames (String prefix, String threads, String time)

Delegates call to BE server for generating the Complete proposal file

The response is wrapped into Single object what allows async call of the BE.

public ResponseEntity<String> reset ()

This method delegates call to BE reset end point

# BE server

## TST

This implementation uses a ternary search trie.

For additional documentation, see <https://algs4.cs.princeton.edu/52trie> Section 5.2 of  
Algorithms, 4th Edition by Robert Sedgewick and Kevin Wayne.

## BEServerController

The controller defines the application end points   
public ResponseEntity<String> getNames(@RequestParam("prefix")String prefix, @RequestParam("threads")String threads, @RequestParam("time") String time)

Invoke the FileProcessService

@GetMapping("/reset")  
public ResponseEntity<String> reset()

Invoke the InputGenerator in order to fill IN folder.

## InputGenerator

This class scans provided word source big file and generates multiple input files in the IN folder

The input file and the result files folder definitions are inside the application.property file and also possible to provide the values from command line the default values are

input.file.name=${user.dir}/DavidCopperField.txt

input.dir=${user.home}/eci/IN

## FileProcessService

This service is an engine of the main functionality

It

1. Build the file list
2. Create the output directory
3. Create the vertx instance
4. Create and Deploy all the Verticles

## Verticles

* TaskPublisherVerticle

Running through the list of files and dispatch the work to FileProcessVerticle using the eventbus

After completion of the file processing it sends message to CompleteSuggestionVerticle in order to generate autoCompleteSuggestion.log file inside the OUT directory

Shutdown vertx at the end of work

* CompleteSuggestionVerticle
* This service starts to process the files inside the OUT folder after receiving notification from TaskPublisherVerticle

At the end of work it notify TaskPublisherVerticle

* FileProcessVerticle

Reading the input file line by line and writing these read lines into file inside OUT folder.

Notify TaskPublisherVerticle about completion