Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

import yargs from "yargs";

import { hideBin } from "yargs/helpers";

import \* as cfg from "./config.js";

import \* as iso from "./iso\_country\_routines.js";

import \* as dbRtns from "./db\_routnies.js";

// Note: hideBin is a shorthand for process.argv.slice(2)

// - bypass the first two arguments

const argv = yargs(hideBin(process.argv))

  .options({

    refresh: {

      demandOption: false,

      describe: "User wants a fresh download of data",

      boolean: true,

    },

    code: {

      demandOption: true,

      describe: "Country code in XX format",

      string: true,

    },

  })

  .help()

  .alias("help", "h")

  .parse();

const baseFunction = async () => {

  try {

    let fileStats = await iso.fileStatsFromFSPromise(cfg.countries);

    let jsonCountries = "";

    let codeQuantity = 0;

    if (fileStats === undefined || argv.refresh) {

      jsonCountries = await iso.getJSONFromWWWPromise(cfg.isocountries);

      let countryList = [];

      await Promise.all(

        jsonCountries.map(async (country) => {

          countryList.push(country);

        })

      );

      codeQuantity = countryList.length;

      await iso.writeFileFromFSPromise(cfg.countries, countryList);

      fileStats = await iso.fileStatsFromFSPromise(cfg.countries);

      console.log("A new " + cfg.countries + " file was written.");

      console.log(cfg.countries + " was created on " + fileStats.birthtime);

      console.log("There are " + codeQuantity + " in " + cfg.countries);

    } else {

      jsonCountries = await iso.getJSONFromWWWPromise(cfg.isocountries);

      let countryList = [];

      await Promise.all(

        jsonCountries.map(async (country) => {

          countryList.push(country);

        })

      );

      codeQuantity = countryList.length;

      let newArr = countryList.map((x) => ({

        name: x.name,

        code: x["alpha-2"],

      }));

      await bulkLoadAndFindByCriteria(newArr);

    }

  } catch (err) {

    console.log(err.message);

  }

};

const bulkLoadAndFindByCriteria = async (inputList) => {

  try {

    const db = await dbRtns.getDBInstance();

    // clean out collection before adding new users

    let results = await dbRtns.deleteAll(db, "countries");

  console.log(

      `there are currently ${results.deletedCount} documents in the countries collection`

    );

    console.log(

      `deleted ${results.deletedCount} documents from countries in the countries collection`

    );

    results = await dbRtns.addMany(db, "countries", inputList);

    console.log(

      `There are now ${results.insertedCount} documents currently in the countries collection`

    );

    let allJEmails = await dbRtns.findAll(

      db,

      "countries",

      { code: argv.code }, // only have addresses contain a j - criteria

      { name: 1, code: 1 } // only return the email field - projection

    );

    allJEmails.forEach((country) => console.log(`The code ${argv.code} belongs to the country of ${country.name}`));

    process.exit(0);

  } catch (err) {

    console.log(err);

    process.exit(1);

  }

};

baseFunction();