package net.fifarm.spider.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.filter.CommonsRequestLoggingFilter;

@Configuration

public class RequestLoggingFilterConfig {

@Bean

public CommonsRequestLoggingFilter logFilter() {

CommonsRequestLoggingFilter filter

= new CommonsRequestLoggingFilter();

filter.setIncludeQueryString(true);

filter.setIncludePayload(true);

filter.setMaxPayloadLength(10000);

filter.setIncludeHeaders(true);

filter.setAfterMessagePrefix("REQUEST DATA : ");

return filter;

}

}

package net.fifarm.spider.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Controller;

import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.context.request.RequestContextHolder;

import org.springframework.web.context.request.ServletRequestAttributes;

import javax.servlet.http.HttpServletRequest;

@Controller

public class IndexController {

private Logger logger = LoggerFactory.getLogger(this.getClass());

@GetMapping("/")

public String goIndex() {

logger.info(String.format("[clientIp] %s", getClientIp()));

return "index";

}

private String getClientIp() {

HttpServletRequest request = ((ServletRequestAttributes) RequestContextHolder.currentRequestAttributes()).getRequest();

String clientIp = request.getHeader("X-Forwarded-For");

if (StringUtils.isEmpty(clientIp)|| "unknown".equalsIgnoreCase(clientIp)) {

clientIp = request.getHeader("Proxy-Client-IP");

}

if (StringUtils.isEmpty(clientIp) || "unknown".equalsIgnoreCase(clientIp)) {

clientIp = request.getHeader("WL-Proxy-Client-IP");

}

if (StringUtils.isEmpty(clientIp) || "unknown".equalsIgnoreCase(clientIp)) {

clientIp = request.getHeader("HTTP\_CLIENT\_IP");

}

if (StringUtils.isEmpty(clientIp) || "unknown".equalsIgnoreCase(clientIp)) {

clientIp = request.getHeader("HTTP\_X\_FORWARDED\_FOR");

}

if (StringUtils.isEmpty(clientIp) || "unknown".equalsIgnoreCase(clientIp)) {

clientIp = request.getRemoteAddr();

}

return clientIp;

}

}

package net.fifarm.spider.controller;

import net.fifarm.spider.service.MongoService;

import org.bson.Document;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/api")

public class JsonController {

@Autowired

MongoService mongoService;

@GetMapping("/search/{name}")

public List<Document> getPlayerListByName(@PathVariable(value="name") String name) {

return mongoService.searchByNameUsingTextScore(name, 100);

}

@GetMapping("/player/{id}")

public Document getPlayerById(@PathVariable(value="id") String id) {

return mongoService.searchById(id, 1).get(0);

}

@GetMapping("/autocomplete")

public List<Document> getPlayerNames(@RequestParam("term") String term) {

return mongoService.searchPlayerNames(term, 30);

}

}

package net.fifarm.spider.controller;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

@Controller

public class PathController {

@GetMapping("/search/{playerName}")

public String searchByName(@PathVariable(value="playerName") String playerName, Model model) {

model.addAttribute("playerName", playerName);

return "playerList";

}

@GetMapping("/player/{playerId}")

public String linkById(@PathVariable(value="playerId") String playerId, Model model) {

model.addAttribute("playerId", playerId);

return "player";

}

}

package net.fifarm.spider.cv;

public class FifarmCV {

public static final String USER\_AGENT = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36";

public static final String FUT\_PLAYER\_API\_URL = "https://www.easports.com/fifa/ultimate-team/api/fut/item?name=%s";

public static final String FUT\_PLAYER\_API\_URL\_BY\_PAGE = "https://www.easports.com/fifa/ultimate-team/api/fut/item?page=%d";

}

package net.fifarm.spider.job.schedule;

import com.google.gson.JsonObject;

import net.fifarm.spider.cv.FifarmCV;

import net.fifarm.spider.net.HttpRequestService;

import net.fifarm.spider.net.Result;

import net.fifarm.spider.service.MongoService;

import net.fifarm.spider.util.JsonUtils;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.stereotype.Component;

import org.thymeleaf.util.StringUtils;

@Component

public class Scheduler {

@Autowired

HttpRequestService httpRequestService;

@Autowired

MongoService mongoService;

private Logger logger = LoggerFactory.getLogger(this.getClass());

@Scheduled(cron = "0 0 1 3 \* ?")

public void getFutItemsFromFifaApi() throws Exception {

int curPage = 1;

String jsonString = requestDataByPage(curPage);

int totalPages = StringUtils.isEmpty(jsonString) ? -1 : getTotalPage(jsonString);

if (totalPages == -1) {

logger.info("GET FUT ITEMS FROM FIFA API IS FAILED");

return;

}

logger.info("GET FUT ITEMS FROM FIFA API -> START");

while (curPage <= totalPages) {

logger.info("GET FUT ITEMS FROM FIFA API -> {}/{}", curPage, totalPages);

mongoService.insertToMongo(requestDataByPage(curPage++));

Thread.sleep(10000); // delay 10 seconds

}

logger.info("GET FUT ITEMS FROM FIFA API -> DONE");

}

private String requestDataByPage(int page) throws Exception {

String url = String.format(FifarmCV.FUT\_PLAYER\_API\_URL\_BY\_PAGE, page);

Result result = httpRequestService.sendGet(url);

if (result.getResponseCode() == 200) {

return result.getResponse();

}

return "";

}

private int getTotalPage(String jsonString) {

JsonObject jsonObject = JsonUtils.getJsonObjectFromString(jsonString);

return Integer.parseInt(JsonUtils.getString(jsonObject, "totalPages"));

}

}

package net.fifarm.spider.net;

import net.fifarm.spider.cv.FifarmCV;

import org.springframework.stereotype.Component;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;

@Component

public class HttpRequestService {

public Result sendGet(String targetUrl) throws Exception {

Result result = new Result();

URL url = new URL(targetUrl);

HttpURLConnection con = (HttpURLConnection) url.openConnection();

con.setRequestMethod("GET");

con.setRequestProperty("User-Agent", FifarmCV.USER\_AGENT);

int responseCode = con.getResponseCode();

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

result.setResponseCode(responseCode);

result.setResponse(response.toString());

return result;

}

}

package net.fifarm.spider.net;

public class Result {

private int responseCode;

private String response;

public int getResponseCode() {

return responseCode;

}

public void setResponseCode(int responseCode) {

this.responseCode = responseCode;

}

public String getResponse() {

return response;

}

public void setResponse(String response) {

this.response = response;

}

@Override

public String toString() {

return "Result{" +

"responseCode=" + responseCode +

", response='" + response + '\'' +

'}';

}

}

package net.fifarm.spider.service;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Service;

@Service

public class LogService {

private final Logger logger = LoggerFactory.getLogger(this.getClass().getSimpleName());

public void trace(String message) {

logger.trace(message);

}

public void debug(String message) {

logger.debug(message);

}

public void info(String message) {

logger.info(message);

}

public void warn(String message) {

logger.warn(message);

}

public void error(String message) {

logger.error(message);

}

}

package net.fifarm.spider.service;

import com.google.gson.JsonArray;

import com.google.gson.JsonElement;

import com.google.gson.JsonObject;

import com.mongodb.DBObject;

import com.mongodb.util.JSON;

import net.fifarm.spider.util.DateUtils;

import net.fifarm.spider.util.JsonUtils;

import org.bson.Document;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Sort;

import org.springframework.data.mongodb.core.MongoTemplate;

import org.springframework.data.mongodb.core.query.Criteria;

import org.springframework.data.mongodb.core.query.Query;

import org.springframework.data.mongodb.core.query.TextCriteria;

import org.springframework.data.mongodb.core.query.TextQuery;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class MongoService {

@Autowired

private MongoTemplate mongoTemplate;

private String playerCollection = "player";

public void insertToMongo(String jsonString) {

JsonArray items = getItems(jsonString);

items.forEach(this::tryInsertItemToMongo);

}

private void tryInsertItemToMongo(JsonElement jEle) {

String id = jEle.getAsJsonObject().get("id").getAsString();

Query query = new Query();

query.addCriteria(Criteria.where("id").is(id));

boolean existsId = mongoTemplate.exists(query, playerCollection);

if (!existsId) {

String now = DateUtils.getNow();

jEle.getAsJsonObject().addProperty("createdDate", now);

jEle.getAsJsonObject().addProperty("updatedDate", now);

mongoTemplate.insert((DBObject) JSON.parse(jEle.toString()), playerCollection);

}

}

private JsonArray getItems(String jsonString) {

JsonObject jsonObject = JsonUtils.getJsonObjectFromString(jsonString);

return JsonUtils.getJsonArray(jsonObject, "items");

}

// db.player.find({$text: {$search: "Heung Min"}},{score:{$meta: "textScore"}}).sort({score:{$meta:"textScore"}}).limit(100)

public List<Document> searchByNameUsingTextScore(String name, int limit) {

TextCriteria textCriteria = TextCriteria.forDefaultLanguage().matching(name);

Query query = TextQuery.queryText(textCriteria).sortByScore().limit(limit);

return mongoTemplate.find(query, Document.class, playerCollection);

}

// db.player.find({"id":"268635560"})

public List<Document> searchById(String id, int limit) {

Query query = new Query();

query.addCriteria(Criteria.where("id").is(id));

query.limit(limit);

return mongoTemplate.find(query, Document.class, playerCollection);

}

// db.player.find({$text: {$search: "Min"}},{"\_id": false, "firstName": true, "lastName": true, "club": true, "nation": true, "headshot": true, "position": true, "composure": true, "quality": true, "id": true}).sort({"composure": -1}).limit(30)

public List<Document> searchPlayerNames(String term, int limit) {

TextCriteria textCriteria = TextCriteria.forDefaultLanguage().matching(term);

Query query = TextQuery.queryText(textCriteria);

query.fields()

.exclude("\_id")

.include("firstName")

.include("lastName")

.include("club")

.include("nation")

.include("headshot")

.include("position")

.include("composure")

.include("quality")

.include("id");

query.with(new Sort(Sort.Direction.DESC, "composure"));

query.limit(limit);

return mongoTemplate.find(query, Document.class, playerCollection);

}

}

package net.fifarm.spider;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication

@EnableScheduling

public class SpiderApplication {

public static final String APPLICATION\_LOCATIONS = "spring.config.location="

+ "classpath:application.yml,"

+ "classpath:real-application.yml";

public static void main(String[] args) {

new SpringApplicationBuilder(SpiderApplication.class)

.properties(APPLICATION\_LOCATIONS)

.run(args);

}

}