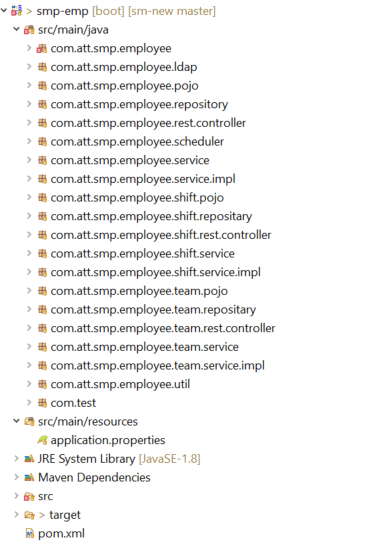
Spring Boot + LdapRetriever + scheduler + Swagger + MongoRepository + Mongodb



Web.xml

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.4.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

< <dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.9</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-mail</artifactId>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.8.0</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.8.0</version>

</dependency>

**Application Properties:**

#spring.data.mongodb.uri=mongodb://localhost:37017/smp

spring.data.mongodb.uri=mongodb://135.52.139.212:27017/employee

#team.mongodb.uri=mongodb://localhost:27017/team

spring.data.mongodb.repositories.type=auto

#temp.filepath = C:/Users/ss392w/Desktop/ShiftManagement/

temp.filepath = /opt/app/softwares/new-igemsqm-csv/

#webphone.data = sattuid=sm496g

#webphone.data = borg=fS030

webphone.data = costcenter=AT18D4220

#webphone.data = attuid=ss392w

#webphone.cron = 5 \* \* \* \* \*

webphone.cron = 0 0 18,19 \* \* SUN

**Field Description Allowed Value**

MIN Minute field 0 to 59

HOUR Hour field 0 to 23

DOM Day of Month 1-31

MON Month field 1-12

DOW Day Of Week 0-6

CMD Command Any command to be executed.

**Ldap Configuration:**

**import** javax.naming.Context;

**import** javax.naming.NamingEnumeration;

**import** javax.naming.NamingException;

**import** javax.naming.directory.Attribute;

**import** javax.naming.directory.Attributes;

**import** javax.naming.directory.DirContext;

**import** javax.naming.directory.InitialDirContext;

**import** javax.naming.directory.SearchControls;

**import** javax.naming.directory.SearchResult;

**public** **class** LdapRetriever {

**private** **static** **final** LdapRetriever ***ldapRetriever***;

**private** **static** String *INITCTX* = "com.sun.jndi.ldap.LdapCtxFactory";

**private** **static** String *MY\_HOST* = "ldap://ldap.webphone.att.com:389";

**private** **static** String *MY\_SEARCHBASE* = "ou=people, o=att, c=us"; //cost card

**static** {

***ldapRetriever*** = **new** LdapRetriever();

}

**private** LdapRetriever() {

}

**public** **static** LdapRetriever getInstance() {

**return** ***ldapRetriever***;

}

**public** List<String> getLdap(String str) {

NamingEnumeration<SearchResult> results = search(str);

String data = getNext(results);

**return** Arrays.*asList*(data.split("!!"));

}

**private** NamingEnumeration<SearchResult> search(String str) {

NamingEnumeration<SearchResult> results = **null**;

String MY\_FILTER = "(" + str + ")";

Hashtable<String, String> env = **new** Hashtable<>();

env.put(Context.***INITIAL\_CONTEXT\_FACTORY***, *INITCTX*);

env.put(Context.***PROVIDER\_URL***, *MY\_HOST*);

**try** {

DirContext ctx = **new** InitialDirContext(env);

SearchControls constraints = **new** SearchControls();

constraints.setSearchScope(SearchControls.***SUBTREE\_SCOPE***);

results = ctx.search(*MY\_SEARCHBASE*, MY\_FILTER, constraints);

} **catch** (NamingException e) {

e.printStackTrace();

}

**return** results;

}

**private** String getNext(NamingEnumeration<SearchResult> results) {

String nextData = "";

**try** {

**while** (results != **null** && results.hasMore()) {

SearchResult si = results.next();

nextData += "" + si.getName();

Attributes attrs = si.getAttributes();

**if** (attrs != **null**) {

NamingEnumeration<? **extends** Attribute> ae = attrs.getAll();

**while** (ae!=**null** && ae.hasMoreElements()) {

Attribute attr = ae.next();

String attrId = attr.getID();

Enumeration<?> vals = attr.getAll();

**while** (vals.hasMoreElements()){

nextData += "|" + attrId + ":" + vals.nextElement();

}

}

nextData+="!!";

}

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** nextData;

}

}

**Scheduled (Cron):**

@EnableScheduling

@Configuration

**public** **class** WebphoneScheduler {

@Autowired

**private** EmployeeService service;

@Autowired

**private** ShiftService shiftService;

@Value("${webphone.data}")

**private** String webphoneData;

// @Scheduled(cron = "0 0 18,19 \* \* SUN") // every sunday 6pm to 7pm

@Scheduled(cron = "${webphone.cron}") // per minute

**public** **void** schedule(){

List<String> list = LdapRetriever.*getInstance*().getLdap(webphoneData);

**for**(**int** i=0;i<list.size()-1;i++){

filter(list.get(i));

}

}

**public** WebphoneScheduler() {

}

**private** **void** filter(String str) {

// System.out.println(str);

String[] data = str.split("\\|");

/\*for(int i=0;i<data.length-1;i++){

System.out.println(data[i]);

}\*/

Pn:”Sachin”|

List<String> list = Arrays.*asList*(data);

String name[] = list.stream()

.filter(line-> line.contains("pn:"))

.findFirst()

.get()

.split(":")[1]

.split(",");

String attuid = list.stream()

.filter(line-> line.contains("attuid:"))

.findFirst()

.get()

.split(":")[1]

.split("\\$")[0];

String designation = list.stream()

.filter(line-> line.contains("jtname:"))

.findFirst()

.get()

.split(":")[1]

.split("\\$")[0];

String consulting = list.stream().filter(line-> line.contains("consulting:") || line.contains("compdesc:")).findFirst().get().split(":")[1].split("\\$")[0];

String telephoneNumber="";

**if**(list.stream().filter(line-> line.contains("telephoneNumber:")).findFirst().isPresent()) {

telephoneNumber = list.stream().filter(line-> line.contains("telephoneNumber:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String mobilePhoneNumber="";

**if**(list.stream().filter(line-> line.contains("mobilePhoneNumber:")).findFirst().isPresent()) {

mobilePhoneNumber = list.stream().filter(line-> line.contains("mobilePhoneNumber:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String mail = "";

**if**(list.stream().filter(line-> line.contains("mail:")).findFirst().isPresent()){

mail = list.stream().filter(line-> line.contains("mail:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String ealias = "";

**if**(list.stream().filter(line-> line.contains("ealias:")).findFirst().isPresent()){

ealias = list.stream().filter(line-> line.contains("ealias:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String sattuid = list.stream().filter(line-> line.contains("sattuid:")).findFirst().get().split(":")[1].split("\\$")[0];

String street = "";

**if**(list.stream().filter(line-> line.contains("street:")).findFirst().isPresent()){

street = list.stream().filter(line-> line.contains("street:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String city = list.stream().filter(line-> line.contains("postalAddress:")).findFirst().get().split(":")[1].split("\\$")[0];

String state = "";

**if**(list.stream().filter(line-> line.contains("st:")).findFirst().isPresent()){

state = list.stream().filter(line-> line.contains("st:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String postalCode = "";

**if**(list.stream().filter(line-> line.contains("postalCode:")).findFirst().isPresent()){

postalCode = list.stream().filter(line-> line.contains("postalCode:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String country = "";

**if**(list.stream().filter(line-> line.contains("friendlyCountryName:")).findFirst().isPresent()){

country = list.stream().filter(line-> line.contains("friendlyCountryName:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String buname = list.stream().filter(line-> line.contains("buname:") || line.contains("bgname:") || line.contains("bdname:")).findFirst().get().split(":")[1].split("\\$")[0];

String bUnit="";

**if**(list.stream().filter(line-> line.contains("bg:")).findFirst().isPresent()) {

bUnit = list.stream().filter(line-> line.contains("bg:")).findFirst().get().split(":")[1].split("\\$")[0];

}

String costcenter = list.stream().filter(line-> line.contains("costcenter:")).findFirst().get().split(":")[1].split("\\$")[0];

/\* Shift Service \*/

Shift shift = **new** Shift();

shift.setId(attuid);

shift.setName(name[1] + " " + name[0]);

shift.setSupervisorAttuid(sattuid);

shift.setTeam(teamList);

shiftService.insertShift(shift);

}

}

**Controller:**

@CrossOrigin(origins = {"http://localhost:4200", "http://m5webappdev02.gcsc.att.com:8000", "https://m5webappdev02.gcsc.att.com:8443"})

@RestController

@RequestMapping("rest/employee")

@Api(value="Employee Data", ~~description~~="operations related to Employee")

**public** **class** EmployeeController {

@Value("${temp.filepath}")

**private** String filePath;

@Autowired

**private** EmployeeService service;

@ApiOperation(value="Add employee data", response=Employee.**class**)

@RequestMapping(path = "insert", method = RequestMethod.***POST***, consumes = MediaType.***APPLICATION\_JSON\_VALUE***, produces = MediaType.***APPLICATION\_JSON\_VALUE***)

**public** Employee insert(@RequestBody Employee employee) {

**return** service.addEmployee(employee);

}

@RequestMapping(path = "update", method = RequestMethod.***POST***, consumes = MediaType.***APPLICATION\_JSON\_VALUE***, produces = MediaType.***APPLICATION\_JSON\_VALUE***)

**public** WriteResult updateByAttId(@RequestBody Employee employee) {

**return** service.update(employee);

}

@RequestMapping(path = "find/{attId}", method = RequestMethod.***GET***, produces = MediaType.***APPLICATION\_JSON\_VALUE***)

**public** Employee findByAttId(@PathVariable("attId") String attId) {

**return** service.findByAttId(attId);

}

}}

**Services:**

**public** **interface** EmployeeService {

**public** Employee addEmployee(Employee employee);

// public Employee updateByAttId(Employee employee);

**public** WriteResult update(Employee employee);

**public** Employee findByAttId(String attId);

**public** List<Employee> getAll();

**public** Page<Employee> getAllEmployeesWithPagiation(Pageable pageable);

**public** List<Employee> search(Employee employee);

**public** Page<Employee> searchWithPagination(Pageable pageable, Employee employee);

**public** List<EmployeeTable> getEmpTable();

}

Service Implementation:

@Service

**public** **class** EmployeeServiceImpl **implements** EmployeeService {

@Autowired

**private** EmployeeRepository repository;

@Override

**public** Employee addEmployee(Employee employee) {

**if**(repository.exists(employee.getAttuid())){

**return** employee;

}**else** {

**return** repository.save(employee);

}

}

@Override

**public** Employee findByAttId(String attId) {

**return** repository.findByAttuid(attId);

}

@Override

**public** List<Employee> getAll() {

**return** repository.findAll();

}

}

**Repository:**

@Repository

**public** **interface** EmployeeRepository **extends** MongoRepository<Employee, String>{

**public** Employee findByAttuid(String attId);

}

**Pojo/Model:**

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.Document;

**import** org.springframework.data.mongodb.core.mapping.Field;

@Document

**public** **class** Employee {

@Id

@Field(value = "attuid")

**private** String attuid;

**private** PersonalData personalData;

**private** LocationData locationData;

**private** AdminData adminData;

**private** OtherUserIds otherUserIds;

**private** CorporateData corporateData;

**public** Employee() {

}

**public** String getAttuid() {

**return** attuid;

}

**public** **void** setAttuid(String attuid) {

**this**.attuid = attuid;

}