链滴

Spring-Ldap

作者：matthewhan

原文链接：https://ld246.com/article/1579138480676

来源网站：链滴

唱歌现在习惯低八度怎么办？(ﾟ﹏ﾟ)

背景

- 经过了这么多年，集团内的各种系统紊乱复杂，结构数据互相同步，而LDAP服务器内的组织架构已没人维护。
- 现有需求需要无纸化办公的组织架构数据增量同步到LDAP服务器中。
- 简单记录下遇到并解决的问题和方法，万一以后又要维护了呢。

LDAP基本概念

LDAP采用目录树的模型，下面是一些概念的解释：

1. 目录树：在一个目录服务系统中，整个目录信息集可以表示为一个目录信息树，树中的每个节点是个条目。

2. 条目：每个条目就是一条记录，每个条目有自己的唯一可区别的名称（DN）。

3. 对象类：与某个实体类型对应的一组属性，对象类是可以继承的，这样父类的必须属性也会被继承来。

4. 属性：描述条目的某个方面的信息，一个属性由一个属性类型和一个或多个属性值组成，属性有必属性和非必须属性。

名词	全称	说明
dc	Domain Component	域名部分，其格式是将完整的域名分成几部分，如域名为example.com变成dc=example,dc=com（一记录的所属位置）。
ou	Organization Unit	组织单元组织单元可以包含多种多个对象（entry），如组织单元名为employee，则ou=employee，那么emp

原文链接：Spring-Ldap

oyee下会有一大堆白给饭桶、精工骨干和划水健将这些实体。

cn Common Name 公共名称，如mc999（一条记录的名称），则cn=hmc999，该条目可以在ou=employee下。

sn Surname 一般用来表达姓，"韩"。

dn Distinguished Name 那上的例子来说，对于hmc999这个实体的定位就是cn=hmc999,ou=employee,dc=example,dc=com当然可能还会有国家、其他多级。

依赖引入

SpringProject已经集成了LDAP组件，直接pom引入即可。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-ldap</artifactId>
</dependency>
```

通过使用Spring LDAP类AttributesMapper和LdapTemplate可以很好地减少传统方式的代码量。

首先我们可以通过AttributesMapper先构建一个实体，用来收集结果，简单样例如下：

```
public class DemoAttributeMapper implements AttributesMapper {
/**
 * 将单个Attributes转成单个对象
 * @param attrs
 * @return
 * @throws NamingException
 */
@Override
public Object mapFromAttributes(Attributes attrs) throws javax.naming.NamingException {

    DemoModel demoModel = new DemoModel();

    /**
     * 通过attr
     */
    if (attrs.get(xxx.name()) != null) {
        demoModel.setXxx(attrs.get(xxx.name()).get().toString());
    }
    ...

    return demoModel;
}
```

编写config类，LDAP服务配置在开发环境的application-dev.properties中，通过注解@Value注入。

```
@Configuration
public class LdapConfiguration {

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

    private LdapTemplate ldapTemplate;
```

```
    @Value("${spring.ldap.urls}")
    private String url;

    @Value("${spring.ldap.base}")
    private String base;

    @Value("${spring.ldap.username}")
    private String userName;

    @Value("${spring.ldap.password}")
    private String passWord;


    @Bean
    public LdapContextSource contextSource() {
        LdapContextSource contextSource = new LdapContextSource();
        Map<String, Object> config = new LinkedHashMap<>(16);

        contextSource.setUrl(url);
        contextSource.setBase(base);
        contextSource.setUserDn(userName);
        contextSource.setPassword(passWord);

        // 解决乱码的关键一句
        config.put("java.naming.ldap.attributes.binary", "objectGUID");

        contextSource.setPooled(true);
        contextSource.setBaseEnvironmentProperties(config);
        log.info(" [ MatthewHan ] : LDAP-Config启动 ");
        return contextSource;
    }

    @Bean
    public LdapTemplate ldapTemplate() {
        if (null == ldapTemplate) {
            ldapTemplate = new LdapTemplate(contextSource());
        }
        return ldapTemplate;
    }
}
```

需要注意的问题

接着就是Ldaptemplate的方法用来增删改查，详细的懒得展开了，有几个点需要注意下：

- Ldaptemplate需要动态生成DN便于增删改查，因为很多操作都是基于构建baseDN，自己需要更实际业务情况，编写工具类用于baseDN的生成。
- 更新操作方法 rebind()，其实是先解绑当前实体，在重新绑定，如果该实体存在下级实体，就会throw错误，所以在不确定条目结构时，可以使用ModifyAttributes类来处理。
- 因为LDAP是树状结构模型，所以在绑定与解绑过程中，一定要注意它是否存在上下级关系，比如在u=department下存在顶级部门、次级部门和下级部门，deId=003001是deId=003的子部门，如果解绑整个部门（包括子部门）那么就不能使用unbind()直接解绑deId=003这个实体，但是可以先解

最下级部门，再一级一级往上解绑。绑定也是一样的原理，先绑定最上级部门，然后再绑定其下级、下级部门。~~题外话：讲道理应该有可以直接解绑该实体以及他的全部子实体吧！但是我好像没找到。。~~

- 遗留的一个问题，害怕会在项目中会引发墨菲定律的风险，已知一个实体的一个属性 deId=003001 包括ou=employee，他的实际DN为deId=003001,deId=003,ou=department...，从DN中我们知他是属于deId=003的子实体，但是可以通过配置AndFilter配置过滤器，从而使用Ldaptemplate的search方法去定位该实体，却不知道他的DN。。我在Ldaptemplate中未找到一个method，这是个未决的问题。关于LDAP的内容，整个互联网上也不是太多，社区也不够活跃，希望以后能够知道答案。

原文链接：Spring-Ldap