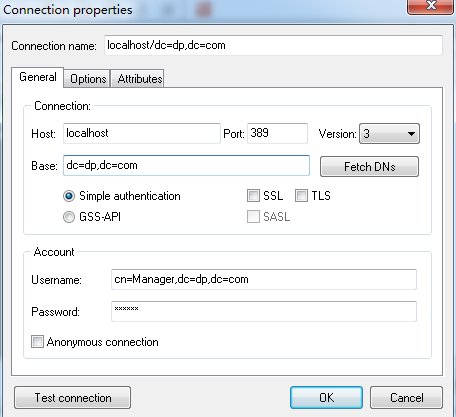
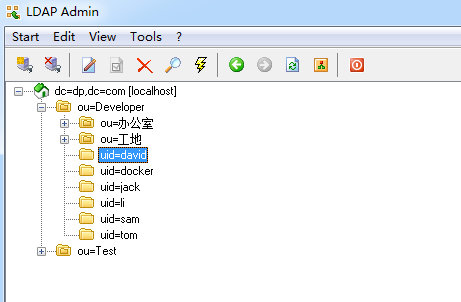
LDAP admin使用说明：

LDAPAdmin 是一个在 Windows 用来编辑 LDAP 账户信息的管理工具，采用 Delphi 开发。和LdapBrowser相比，LDAPAdmin可以功能更全面，除了查看还可以添加修改已有的LDAP账户数据。首先使用LDAPAdmin连接到本地LDAP Server，注意要使用账号密码连接，否则修改数据时会报权限不足的错误。

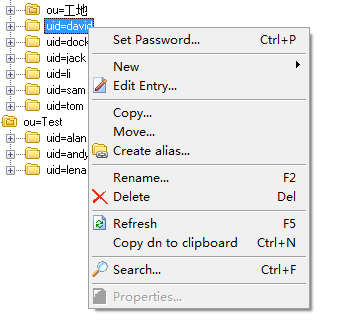
LDAP Server默认的账号密码为Manager / secret



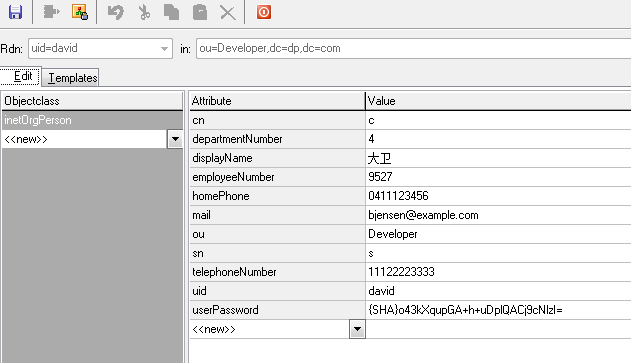
连接成功后就可以看到LDAP的数据了

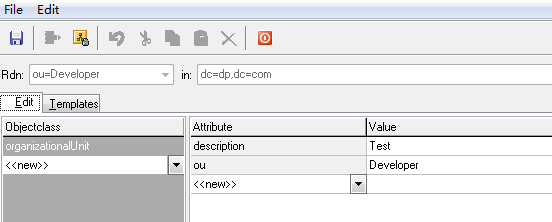


其中dc=dp,dc=com表示连接的LDAP“数据库”的名称；ou表示部门，当前机构下，有Developer和Test两个一级部门，在Developer部门下还有两个二级部门分别为办公室和工地。单击uid=david表项，点击右键可以查看该项的详细信息并且进行修改。



LDAP常用attribute





attribute的名称是固定的，不能自己随便定义，用户部门常用的attribute如上图所示

接下来看一看ObjectClass与Attribute的关系。  
如同Java里面的一个类可以包括多个Field，在业务上可能会定义某些Field是必须的，

另外一些是可选的。在LDAP中也存在类似关系，每一个 ObjectClass都定义了一些Attribute。

在这些Attriubte中分为两种类型MUST，MAY， MUST表示这个Entry必须包括的属性，MAY

为可选。一个ObjectClass的Attribute也包括所有继承自父ObjectClass和自身定义的

ObjectClass。

objectClass有着严格的等级之分，最顶层是top和alias。例如，organizationalPerson

这个objectClass就隶属于person，而person又隶属于top。

objectClass可分为以下3类：   
结构型（Structural）：如person和organizationUnit；   
辅助型（Auxiliary）：如extensibeObject；   
抽象型（Abstract）：如top，抽象型的objectClass不能直接使用。

**Java连接LDAP**

public class LDAP {  
 public static DirContext getCtx() {  
 DirContext ctx = null;  
 String account = "Manager";   
// String password = "secret";   
 String root = "dc=dp,dc=com";   
 Hashtable env = new Hashtable();  
 env.put(Context.*INITIAL\_CONTEXT\_FACTORY*, "com.sun.jndi.ldap.LdapCtxFactory");  
 env.put(Context.*PROVIDER\_URL*, "ldap://localhost:389/" + root);  
 env.put(Context.*SECURITY\_AUTHENTICATION*, "none");  
 env.put(Context.*SECURITY\_PRINCIPAL*, "cn="+account );  
// env.put(Context.SECURITY\_CREDENTIALS, password);  
 try {  
 // 链接ldap  
 ctx = new InitialDirContext(env);  
 System.*out*.println("认证成功");  
 } catch (javax.naming.AuthenticationException e) {  
 System.*out*.println("认证失败");  
 } catch (Exception e) {  
 System.*out*.println("认证出错：");  
 e.printStackTrace();  
 }  
 return ctx;  
 }  
  
 //暂未使用  
 public static boolean verifySHA(String ldappw, String inputpw)  
 throws NoSuchAlgorithmException {  
  
 // MessageDigest 提供了消息摘要算法，如 MD5 或 SHA，的功能，这里LDAP使用的是SHA-1  
 MessageDigest md = MessageDigest.*getInstance*("SHA-1");  
  
 // 取出加密字符  
 if (ldappw.startsWith("{SSHA}")) {  
 ldappw = ldappw.substring(6);  
 } else if (ldappw.startsWith("{SHA}")) {  
 ldappw = ldappw.substring(5);  
 }  
  
 // 解码BASE64  
 byte[] ldappwbyte = Base64.*decode*(ldappw);  
 byte[] shacode;  
 byte[] salt;  
  
 // 前20位是SHA-1加密段，20位后是最初加密时的随机明文  
 if (ldappwbyte.length <= 20) {  
 shacode = ldappwbyte;  
 salt = new byte[0];  
 } else {  
 shacode = new byte[20];  
 salt = new byte[ldappwbyte.length - 20];  
 System.*arraycopy*(ldappwbyte, 0, shacode, 0, 20);  
 System.*arraycopy*(ldappwbyte, 20, salt, 0, salt.length);  
 }  
  
 // 把用户输入的密码添加到摘要计算信息  
 md.update(inputpw.getBytes());  
 // 把随机明文添加到摘要计算信息  
 md.update(salt);  
  
 // 按SSHA把当前用户密码进行计算  
 byte[] inputpwbyte = md.digest();  
  
 // 返回校验结果  
 return MessageDigest.*isEqual*(shacode, inputpwbyte);  
 }  
  
 public static void main(String[] args) {  
 DirContext ctx = null;  
 boolean flag=false;  
 String name = "Tom c";  
 try {  
 ctx = *getCtx*();  
 SearchControls constraints = new SearchControls();  
// constraints.setSearchScope(SearchControls.ONELEVEL\_SCOPE);  
// constraints.setSearchScope(SearchControls.OBJECT\_SCOPE);  
 constraints.setSearchScope(SearchControls.*SUBTREE\_SCOPE*);  
 NamingEnumeration en = ctx.search("", "uid=" + "Tom", constraints); // 查询所有用户  
 while (en != null && en.hasMoreElements()) {  
 Object obj = en.nextElement();  
 if (obj instanceof SearchResult) {  
 SearchResult si = (SearchResult) obj;  
 System.*out*.println("name: " + si.getName());  
 Attributes attrs = si.getAttributes();  
 if (attrs == null) {  
 System.*out*.println("No attributes");  
 } else {  
 Attribute attr = attrs.get("userPassword");  
 Object o = attr.get();  
 byte[] s = (byte[]) o;  
 String pwd2 = new String(s);  
// flag = verifySHA(pwd2, "123456");  
 if(pwd2.equals("123456")){  
 flag = true;  
 }  
 }  
 } else {  
 System.*out*.println(obj);  
 }  
 System.*out*.println();  
 }  
 ctx.close();  
 } catch (NamingException ex) {  
 try {  
 if (ctx != null) {  
 ctx.close();  
 }  
 } catch (NamingException namingException) {  
 namingException.printStackTrace();  
 }  
 }  
 if(flag){  
 System.*out*.println("验证通过");  
 }  
 }  
  
}