Testing LDAP Implementations

Emmanuel Lécharny
Do who need tests anyway?

OSS projects don't need it...

We have users!
YOU'RE DOING IT WRONG
LDAP project phases

- Initial analysis
- Development + tests
- Conformance tests

Tests are costly, and must be run frequently...
LDAP Tests

• Unit tests
• Integration tests
• Performance tests
You need some tests yo!
Unit Tests in Java

• Need a server we can launch
• Need an API
• More than that, need some mechanism to speed up tests
ApacheDS test framework

• We can start a server using annotations
• We provide an easy to use API
• Tests can be run concurrently
• No need to start/stop or cleanup the server for each test
Simple test

• Creation of a DirectoryService
• Creation of a LdapServer
• Extends AbstractLdapTestUnit
• Get an LdapConnection
• And now we can send requests...
@RunWith(FrameworkRunner.class) // Define the DirectoryService
@CreateDS() // Define the LDAP protocol layer
@CreateLdapServer(
    transports =
    {
        @CreateTransport(protocol = "LDAP")
    })

public class A_SimpleServerTest extends AbstractLdapTestUnit
{
    /**
     * A simple test
     */
    @Test
    public void test() throws Exception
    {
        LdapServer ldapServer = getLdapServer();

        // Get an admin connection on the defined server
        LdapConnection connection = new LdapNetworkConnection("localhost", ldapServer.getPort());
        connection.bind("uid=admin,ou=system", "secret");

        // Check that we can read an entry
        assertNotNull(connection.lookup("ou=system"));

        // And close the connection
        connection.close();
    }
}
Test with entries injection

• Same as the previous example
• Injection of entries with `@ApplyLdifs` or `@ApplyLdifFiles`
@ApplyLdifs()
{
    // Entry # 1
    "dn: uid=elecharny,ou=users,ou=system",
    "objectClass: uidObject",
    "objectClass: person",
    "objectClass: top",
    "uid: elecharny",
    "cn: Emmanuel Lécharny",
    "sn: lecharny",
    "userPassword: emmanuel"
}

@CreateDS()
@CreateLdapServer()
    transports =
    {
        @CreateTransport(protocol = "LDAP")
    }

public class B_LdifEntryServerTest extends AbstractLdapTestUnit
{
    /**
     * A test where we bind using the added entry credentials
     */
    @Test
    public void testBindUser() throws Exception
    {
        // Get a connection (not bound yet) on the server
        LdapConnection connection = getWiredConnection( getLdapServer() );
        connection.bind( "uid=elecharny,ou=users,ou=system", "emmanuel" );

        // Check that we can read an entry
        assertNotNull( connection.lookup( "uid=elecharny,ou=users,ou=system" ) );

        // And close the connection
        connection.close();
    }
}
Test with partition creation

- Creation of a DirectoryService
  - Creation of a Partition
    - Creation of indexes
  - Etc...
@RunWith(FrameworkRunner.class)  // Define the DirectoryService
@CreateDS(
    partitions =
    {
        @CreatePartition(
            name = "example",
            suffix = "dc=example,dc=com",
            contextEntry = @ContextEntry(
                entryLdif =
                "dn: dc=example,dc=com\n" +
                "dc: example\n" +
                "objectClass: top\n" +
                "objectClass: domain\n" ),
            indexes =
            {
                @CreateIndex( attribute = "objectClass" ),
                @CreateIndex( attribute = "dc" ),
                @CreateIndex( attribute = "ou" )
            } )
    }
)
@CreateLdapServer(  // Define the LDAP protocol layer
    transports =
    {
        @CreateTransport(protocol = "LDAP")
    }
)
public class D_ServerWithPartitionTest extends AbstractLdapTestUnit
{
    @Test
    public void test() throws Exception
    {
        // Get an admin connection on the defined server
        LdapConnection connection = getWiredConnection( getLdapServer(), "uid=admin,ou=system", "secret" );

        // Check that we can read the Example context entry
        assertNotNull( connection.lookup( "dc=example,dc=com" ) );
...

Saving start/stop delays

• No need to start a fresh server for each test
• No need to revert the modifications when the test is done
• Automatic rollback
• OTOH, kills concurrent tests...
Defining more than one server

• May be needed
• Can be associated to a suite, a class or a method
Modifying the schema

- Easy to modify
- Use `@ApplyLdif$` or `@ApplyLidfFiles` for that purpose
- Will be reverted when the test will end, as usual
JMeter

- User friendly GUI
- Tests can be exported and executed
- Remote agents can be used
- No code needed
INNOVATION

If it can make your job easier, it can probably make it irrelevant.
API

• Schema aware
• Easy to use
• Deal locally with comparisons
@ApplyLdifs(
    
    // Entry # 1
    "dn: cn=Test Lookup,ou=system",
    "objectClass: person",
    "cn: Test Lookup",
    "sn: sn test"
)

public void testLookupCn() throws Exception {

    LdapConnection connection = getWiredConnection( getLdapServer(), "uid=admin,ou=system", "secret" );
    Entry entry = connection.lookup( "cn=test lookup,ou=system", "cn" );
    assertNotNull( entry );

    // Check that we don't have any operational attributes:
    // We should have only 3 attributes: objectClass, cn and sn
    assertEquals( 1, entry.size() );

    // Check that all the user attributes are present
    assertTrue( entry.contains( "cn", "Test Lookup" ) );
    assertFalse( entry.contains( "cn", "test lookup" ) );
    assertFalse( entry.contains( "2.5.4.3", "test lookup" ) );
    assertFalse( entry.contains( "CN", " test  LOOKUP  " ) );

}
```java
@ApplyLdifs{
    // Entry # 1
    "dn: cn=Test Lookup,ou=system",
    "objectClass: person",
    "cn: Test Lookup",
    "sn: sn test"
}

public void testLookupCn() throws Exception {
    LdapConnection connection = getWiredConnection(
        getLdapServer(), "uid=admin,ou=system", "secret")
    ;

    // Make the connection schema aware
    connection.loadSchema();

    Entry entry = connection.lookup("cn=test lookup,ou=system", "cn");
    assertNotNull(entry);

    // Check that we don't have any operational attributes :
    // We should have only 3 attributes : objectClass, cn and sn
    assertEquals(1, entry.size());

    // Check that all the user attributes are present
    assertTrue(entry.contains("cn", "Test Lookup"));
    assertTrue(entry.contains("cn", "test lookup"));
    assertTrue(entry.contains("2.5.4.3", "test lookup"));
    assertTrue(entry.contains("CN", "  test    LOOKUP  ");
}
```
The Future
NEXT EXIT
Future

• Use Studio to register scenarii
• 'Reboot' Slamd effort (or design a new tool)
• Provide a Groovy LDAP API
• Add LDAP assertions
• Make the Java tests able to start another server
• LDAPUnit : a dedicated LDAP test framework
Thanks !