

LDAP & Cocoon

Real life, down-to-earth Cocoon and LDAP

a talk by :

Ross McDonald
&
Jeremy Quinn
@
CocoonGT 2006

*I want you to believe every word I say
I want you to believe every thing I do*

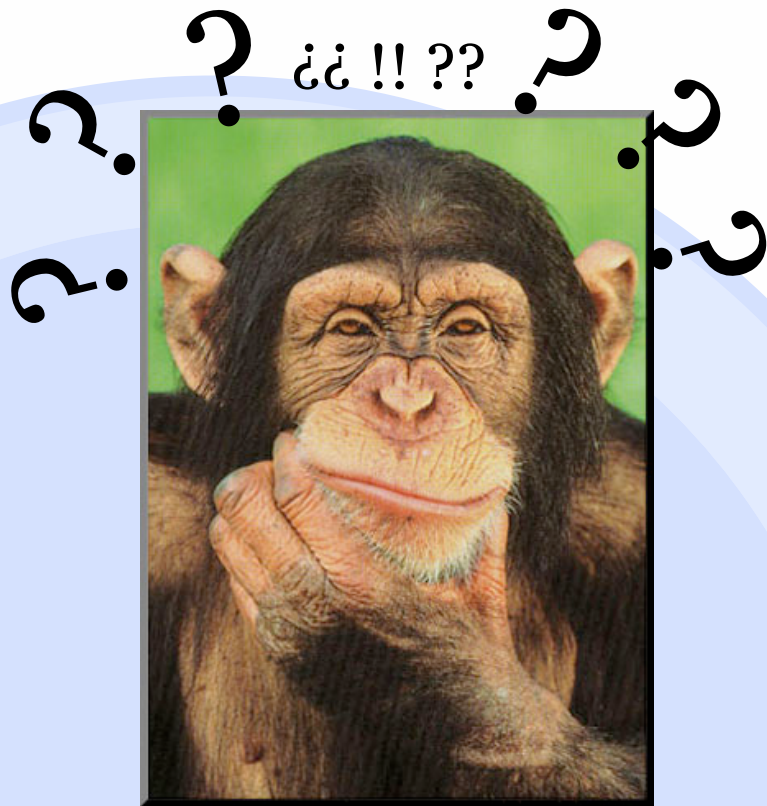
Toots and the Maytals



COCOON

Wait for the audience to calm down :-)

Why this talk ?



LDAP The missing manual

Ross: Can you guess who is who ?

Jeremy: Why are we doing this talk? (because Ross cried like a baby when he saw the state of the docs)

Ross: it is geared to people like us.. who want to create an 'enabling' framework, so others can get on with the work, it does not take a vastly experienced cocoon developer to do this, when there are well documented examples

Ross: How are we qualified? We have both built registration systems using LDAP and Cocoon

Ross: We are just typical users ... now that we have fought our way to an implementation, we want to make it easy for others. And it never hurts to have a little more documentation around!

○ ● ● What is LDAP

- **Lightweight Directory Access Protocol**
- **A standard for accessing directory services over networks**
- **A modern lightweight successor to X500, WHOIS etc etc**

Very brief definition of what LDAP is

Ross: LDAP stands for “Lightweight Directory Access Protocol”

Jeremy: LDAP is not a database, it is a protocol for accessing directory information

Ross: It turns out that LDAP is really not as complicated as you might first expect, my understanding and opinion has totally changed on it over the last four months.. initially I was very skeptical.. now I have found myself championing it in the workplace!

When and why

should I use LDAP ?

- Hierarchical Data
- Distributed Data
- Store Almost Anything
- Fast, Scalable, Searchable

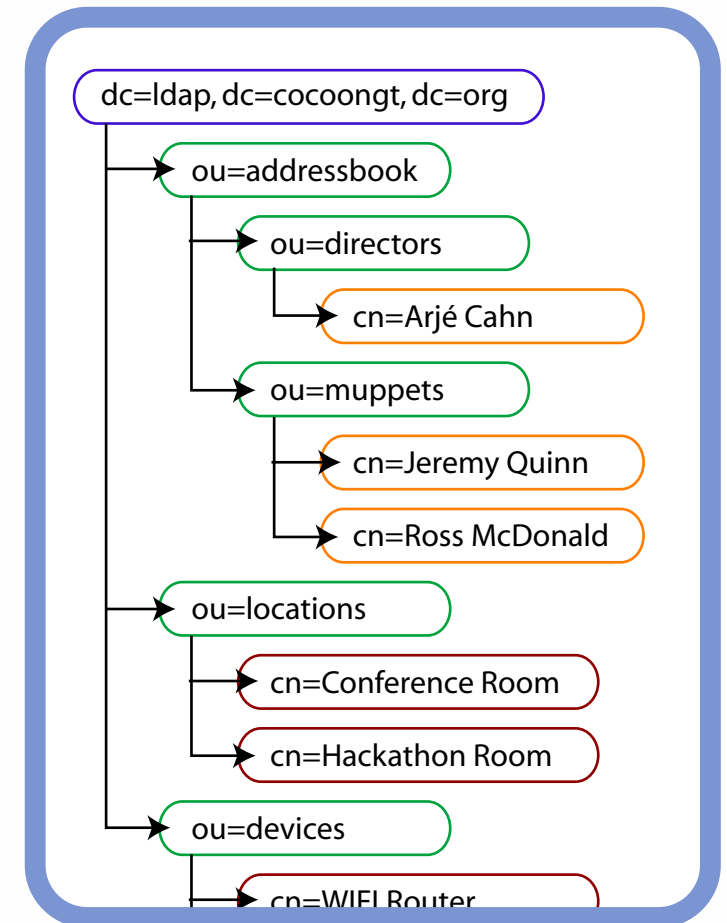
4

Ross: Just launched a project with a massive emphasis on LDAP in collaboration with Sourcesense, they provided the LDAP and data setup, we had to write the code to integrate it into our websites
Ross: It turns out LDAP is very good at authentication, and personalisation

Jeremy: when I worked for Luminas, we used LDAP on a project for a big university etc.

Hierarchical Data

- An address book comprising multiple departments
- Departments comprising multiple contact entries
- Contact entries could contain more hierarchy

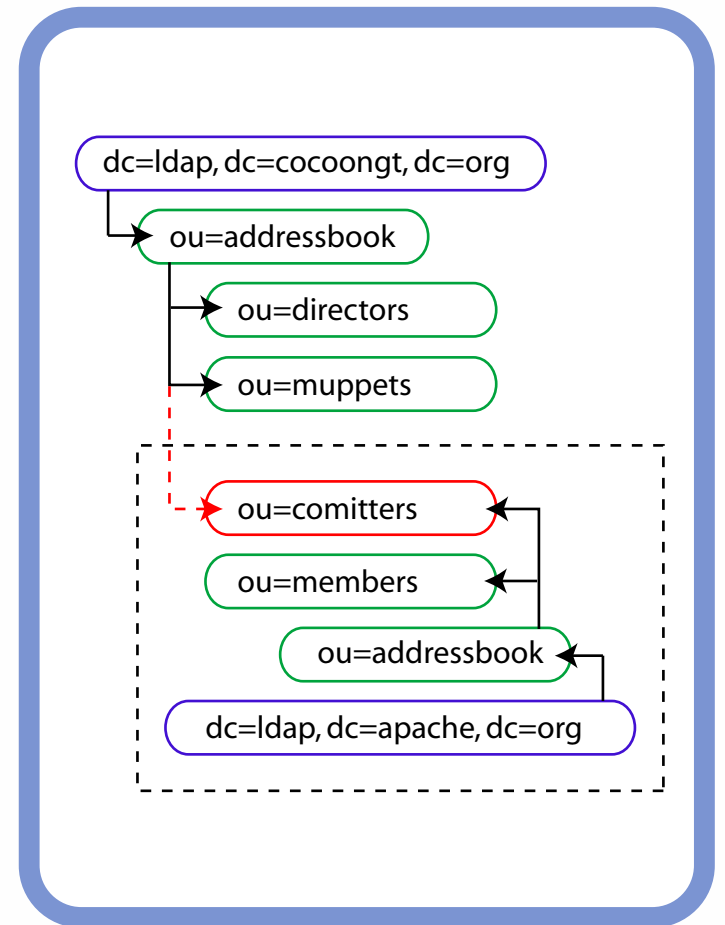


Ross: its just not fun creating and maintaining hierarchies in RDB

Ross: In our app we chose to use multi-level data because this is something worth documenting within LDAP, it is not necessarily intuitive... In the course of our most recent project, I had to explain the workings of LDAP and working with its data to several colleagues, ranging from project managers to designers, and understanding LDAP itself proved the biggest hurdle... Later on, we will look at how simple it is in fact to work with structured data.

Distributed Data

- Aggregate datasources
- Reference external sources
- Load Balance
- Replicate automatically

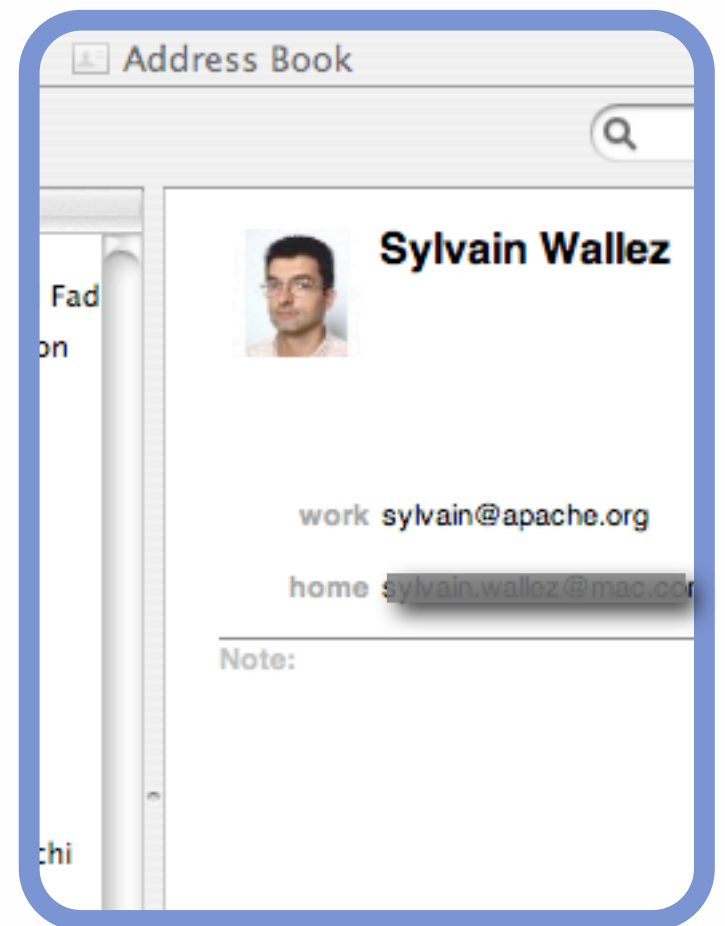


Jeremy: you can join different info sources together, here is a conceptual example of adding the ou=comitters group to our addressbook

Ross: easier replication than MySQL, which forces storage to be in-memory

Store Almost Anything

- UTF-8 Strings, Booleans, Numbers, Phone Numbers, Dates, Email Addresses, Postal Addresses, Certificates, Passwords, Image, Audio, References, Directory Paths, URLs, Countries, etc. etc.
- Multivalued Attributes



Mean Machine

- Fast
- Scalable
- Searchable
- Standards-based

Searching

```
BASE      : Where?
SCOPE     : How deep?
FILTER    : Find What?
           { equality
             presence
             substring
             < and/or >
             like
             etc. }
RETURN   : Which Attrs?
```

Ross: easy access from a given node instantly, no need to do lots of Table joins

Ross: We had a 23gig instance for just our UK data

Jeremy: LDAP directories are heavily optimized for read performance

Jeremy: Above is an example of the kind of criteria you can use to describe a search

Ross: LDAP is such an open standard, and now with SyncML data may be read from many different devices, for gadget freaks like me, thats fun! You can have one central contacts and calendaring system, synchronising to all your devices.

○ ● ● What LDAP is NOT

- Not well documented
- Not a replacement for SQL
- Not so good for dynamic data

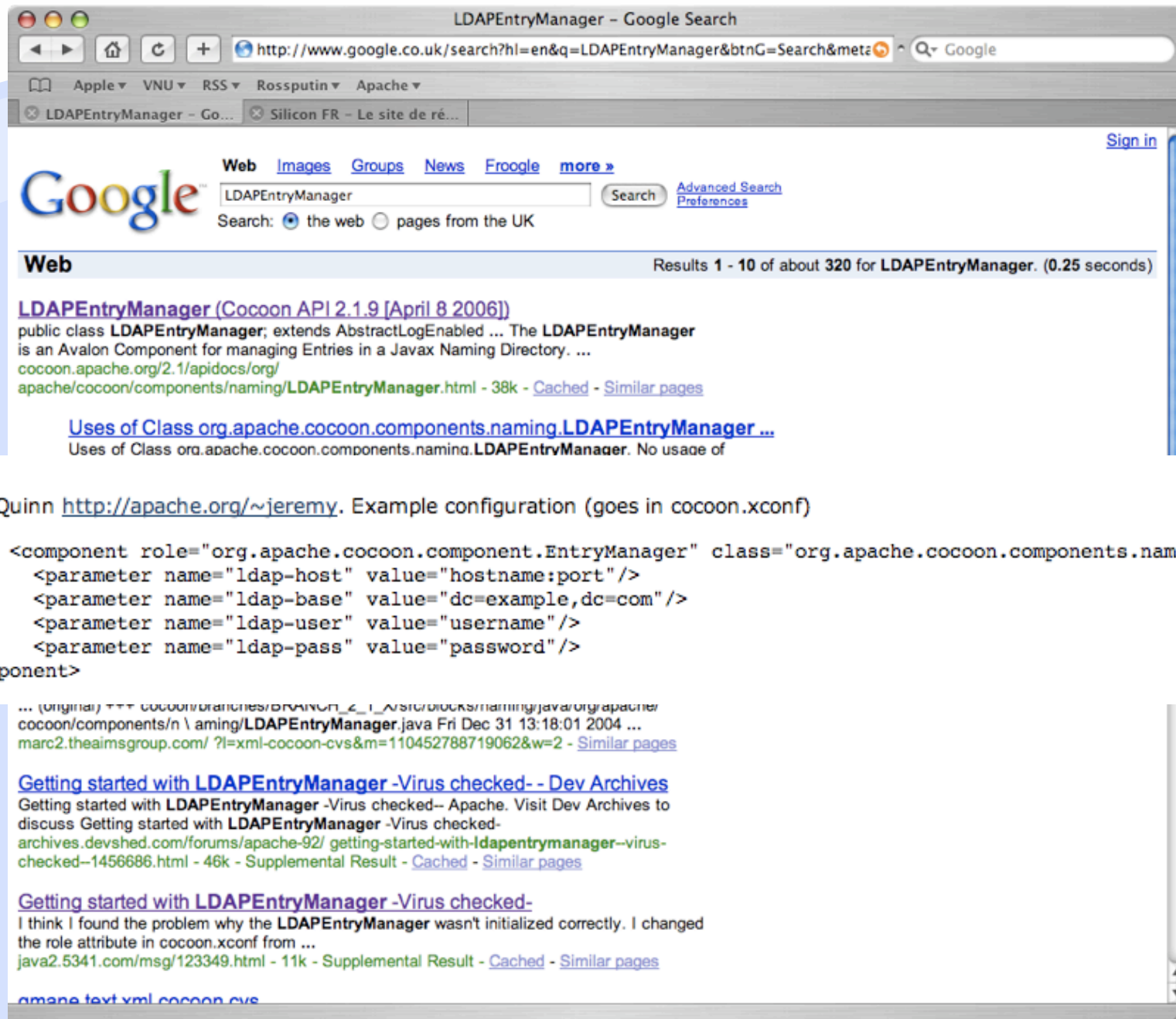
Quickly dispel some myths

Ross: It is more difficult to setup than an RDBs

Jeremy: Hang on, it is much easier than that !!

Ross: Only because I worked it out and told you how, you feckwit :-)

Examples of Googling



The screenshot shows a Google search for "LDAPEntryManager". The search results include a link to the Cocoon API documentation, a link to a configuration example by Jeremy Quinn, and several forum posts discussing the class. The first result is "LDAPEntryManager (Cocoon API 2.1.9 [April 8 2006])" with a description: "public class LDAPEntryManager; extends AbstractLogEnabled ... The LDAPEntryManager is an Avalon Component for managing Entries in a Javax Naming Directory. ...". Below this is a code block for an XML configuration snippet.

Author:
Jeremy Quinn <http://apache.org/~jeremy>. Example configuration (goes in cocoon.xconf)

```
<component role="org.apache.cocoon.component.EntryManager" class="org.apache.cocoon.components.naming.LDAPEntryManager"
  <parameter name="ldap-host" value="hostname:port" />
  <parameter name="ldap-base" value="dc=example,dc=com" />
  <parameter name="ldap-user" value="username" />
  <parameter name="ldap-pass" value="password" />
</component>
```

Just a quick example of what was available...

Ross: So unfortunately, the web was lacking a beginning to end example, and what there was, had some inaccuracies.. don't worry, we fixed up the javadocs!

○ ● ● A Contrived Example

create remove update delete

- Sample CRUD Application
- JXT + CForms + Flowscript
- Uses LDAPEntryManager
- Written in one day

Getting Started

why might this be perceived as difficult ?

- Define your Structure
- Configure your Server
- Populate your Server
- Alphabet Soup

Ross: lack of documentation
Jeremy: SORRY !!!! ;)

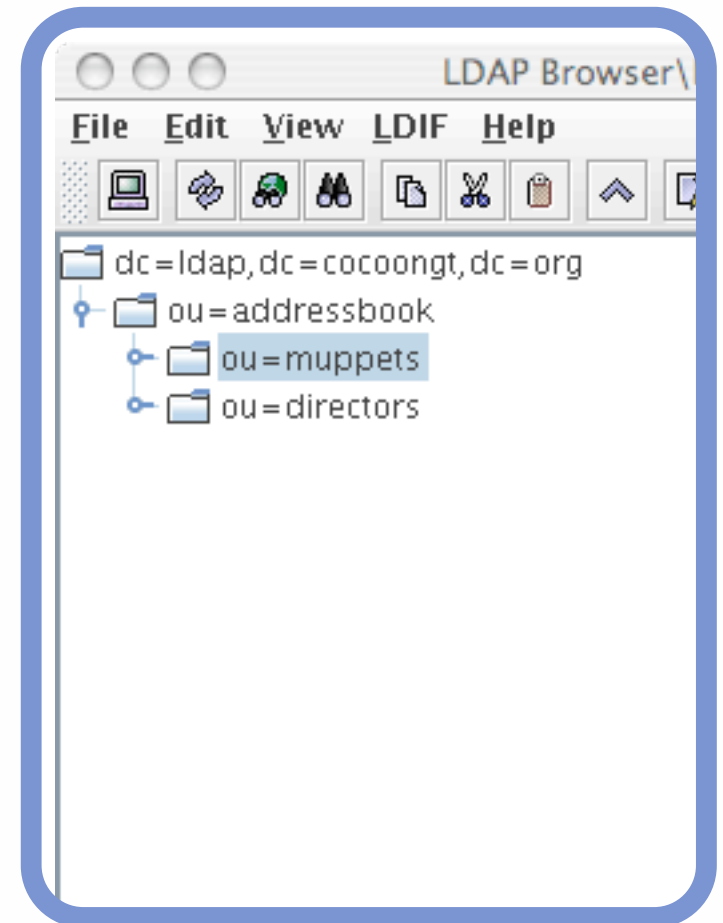
Define your Structure

```
dn: dc=ldap,dc=cocoongt,dc=org
objectClass: top
objectClass: dcObject
objectClass: organization
dc: cocoongt
o: Cocoon GetTogether LDAP
```

```
dn: ou=addressbook,dc=ldap,dc=cocoongt,dc=org
objectClass: top
objectClass: organizationalUnit
ou: addressbook
```

```
dn: ou=muppets,ou=addressbook,dc=ldap,dc=cocoongt,dc=org
objectClass: top
objectClass: organizationalUnit
ou: muppets
```

```
dn: ou=directors,ou=addressbook,dc=ldap,dc=cocoongt,dc=org
objectClass: top
objectClass: organizationalUnit
ou: directors
```



Jeremy: Many pre-defined Schemas, DataTypes and Attributes for specific tasks
Ross: We chose a standard schema that enables connection to multiple remote entities with no configuration...Email Clients, addressbooks

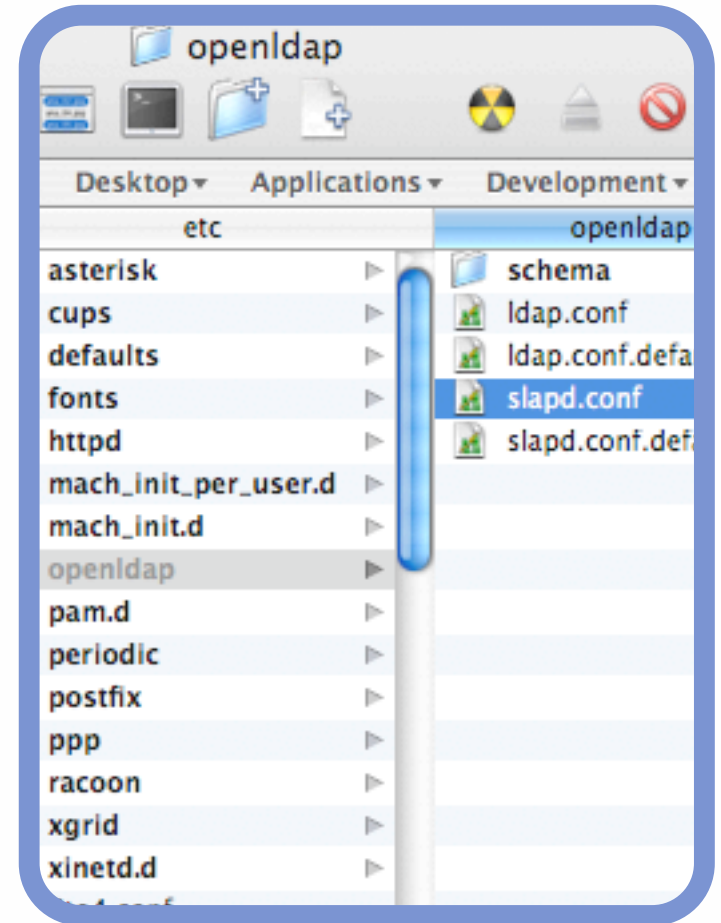
Configure your Server

```
##
# slapd.conf file for Addressbook Sample
##

include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/nis.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/misc.schema
include /etc/openldap/schema/samba.schema
include /etc/openldap/schema/apple.schema
pidfile /var/run/slapd.pid
argsfile /var/run/slapd.args
allows bind_v2
schemacheck off

database bdb
directory /var/db/openldap/openldap-data/
suffix "dc=ldap,dc=cocoongt,dc=org"
rootdn "dc=ldap,dc=cocoongt,dc=org"

# MD5 hash of 'secret'
rootpw {SSHA}G0u19DgKDPWkoQnxIQV6VNvuIu8Bfboz
```



14

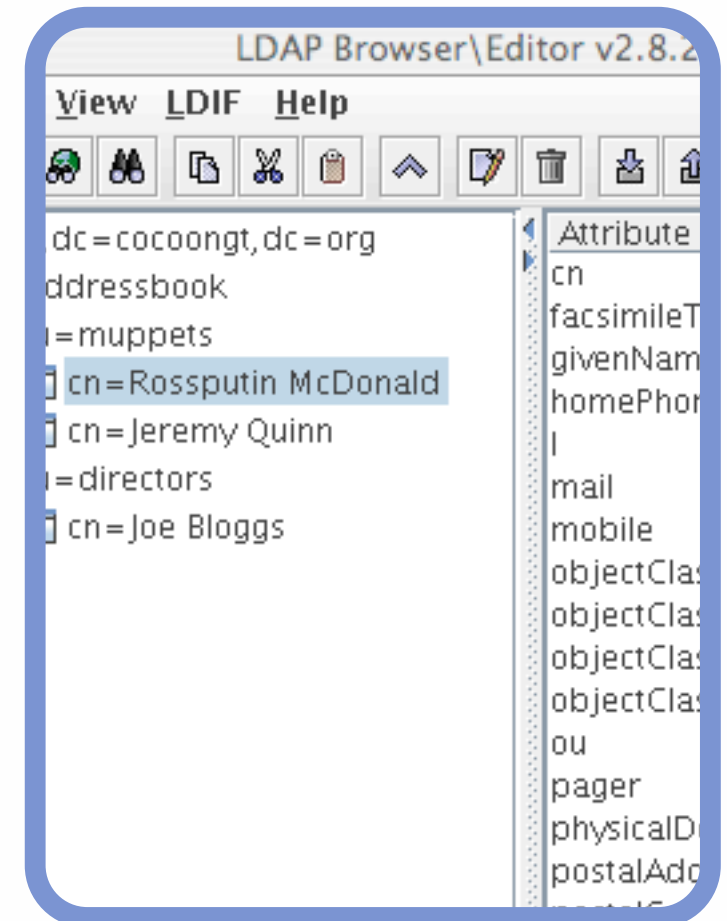
Jeremy: This is from the sample App we provide, it is a setup for MacOSX that has OpenLDAP already installed but not running.

Ross: Once again, things are never straight forward, after some research you eventually realise that the mac instance of openldap is configured to dislike plain text passwords :-)

Populate your Server

```
# sample data to fill the addressbook
```

```
dn: cn=Rossputin McDonald, ou=muppets, ou=addressbook, dc=ldap, dc=cocoongt, dc=org
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Rossputin McDonald
gn: Rossputin
sn: McDonald
mail: rossputin@rossputin.org
physicalDeliveryOfficeName: Cocoon GetTogether LDAP
postalAddress: PO BOX 55555
l: Newbury
st: Berkshire
postalCode: RG19 4QL
telephoneNumber: 666
facsimileTelephoneNumber: 666
pager: 666
mobile: 666
homePhone: 666
ou: muppets
```



```
dn: cn=Jeremy Quinn, ou=muppets, ou=addressbook, dc=ldap, dc=cocoongt, dc=org
objectClass: top
objectClass: person
objectClass: organizationalPerson
```

15

Jeremy: This is some sample data from the App we provide, to pre-populate the repository
Jeremy: Here we use the LDIF – text format, you may also use the XML format – DSML

Alphabet Soup

Many Attributes (RFC 1779)

cn : Common Name
gn : Given Name
sn : Surname
mail : Email Address
o : organisation
physicalDeliveryOfficeName : Department Name
postalAddress : Address Line 1
l : Address Line 2
st : Address Line 3
postalCode : Post Code
c : Country
telephoneNumber : Office Number
facsimileTelephoneNumber : Fax Number
pager : Pager Number
mobile : Mobile Number
homePhone : Home Number
ou : Organisational Unit
dc : Domain Component
dn : Distinguished Name



X.500 Varieties

Jeremy: There are many pre-defined Attributes, they come from X.500
Ross: How many designers and developers looking at LDAP for the first time are going to make anything meaningful of all this ?

LDAP Component

using it from your own code

- Which Component should I use?
- Adding the Naming Block
- Adding the LDAPEntryManager
- Code Hierarchy
- Calling the LDAPEntryManager

Ross: Discuss the merits of the different options in Cocoon ? After some research, and a chat with Jeremy, we chose to use the LDAPEntryManager, as it offered a more flexible solution

Jeremy: runs through the setup

○ ● ● The Naming Block

local.blocks.properties

```
#-----[dependency]: "itext" depends on "xsp" (for samples).
include.block.itext=false
include.block.jfor=false
include.block.jsp=false
#-----[dependency]: "linkrewriter" depends on "xsp".
include.block.linkrewriter=false
#-----[dependency]: "lucene" is needed by "querybean".
include.block.lucene=false
include.block.midi=false
```

#include.block.naming=false

```
#-----[dependency]: "obj" depends on "databases" (for samples), "forms" (for samples),
"hsqldb" (for samples), "xsp" (for samples).
#-----[dependency]: "obj" is needed by "javaflow", "portal", "querybean".
include.block.obj=false
include.block.paranoid=false
include.block.poi=false
```

now recompile

LDAPEntryManager

```
<component
  role="org.apache.cocoon.components.naming.EntryManager"
  class="org.apache.cocoon.components.naming.LDAPEntryManager"
  logger="flow.ldap">

  <parameter name="ldap-host" value="ldap://localhost:389"/>
  <parameter name="ldap-base"
    value="ou=addressbook,dc=ldap,dc=cocoongt,dc=org"/>
  <parameter name="ldap-user" value="dc=ldap,dc=cocoongt,dc=org"/>
  <parameter name="ldap-pass" value="secret"/>

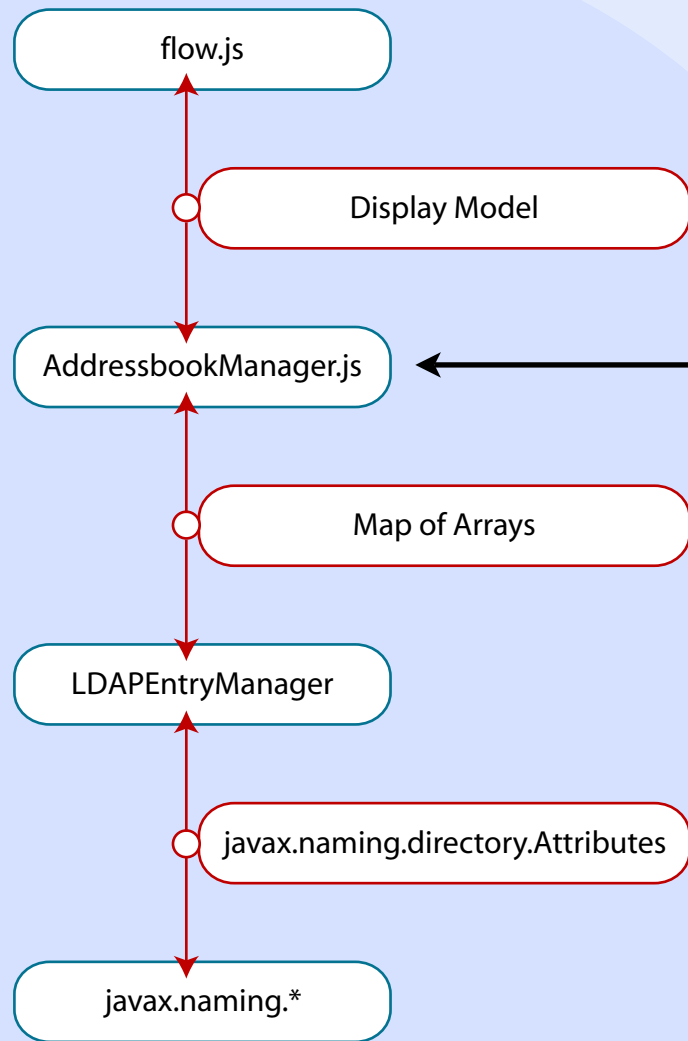
</component>
```

Add the Component to cocoon.xconf

Setting up the Component

explain the base context, we do not need to use ou=addressbook inside the app

Code Hierarchy



```
var AddressbookManager = {  
  binding: {  
    firstname : "givenName",  
    lastname  : "sn",  
    email     : "mail",  
    deptName  : "physicalDeliveryOfficeName",  
    address1  : "postalAddress",  
    address2  : "l",  
    address3  : "st",  
    postcode  : "postalCode",  
    ophone    : "telephoneNumber",  
    fphone    : "facsimileTelephoneNumber",  
    pager     : "pager",  
    mphone    : "mobile",  
    hphone    : "homePhone",  
    dept      : "ou"  
  }  
}
```

Flow layer, talks to AddressBook Application Layer, which talks to Component Application layer converts between Display Names and LDAP Attribute Names
The LDAPEntryManager provides an extremely simplified set of common LDAP Methods and Data Types, to simplify access from FlowScript

Get, Use, Dispose

```
function getPerson() {
  var uid = cocoon.parameters["dn"];
  var entrymanager = cocoon.getComponent(EntryManager.ROLE);
  try {
    var person = AddressbookManager.getPerson(
      entrymanager, uid
    );
    cocoon.sendPage(cocoon.parameters["screen"], {person: person});
  } catch (e) {
    print(e);
    throw(e);
  } finally {
    cocoon.releaseComponent(entrymanager);
  }
}
```

21

Ross: So.. how do we use Jeremy's component within Cocoon? Flowscript!

First we create an instance of the LDAPEntryManager component. Then we attempt to use it in a try catch block, before releasing resources in a 'finally' clause

Easy right? Why then have we pushed this part of the talk towards the end, into the intermediate section?

Well flowscript is easy, its javascript... but what about those objects we get back from calls to Java.. our experience showed that many developers found it hard to know what to expect, sometimes an Array, sometimes a hashmap... Also, there is often a lack of structure in javascript, and spaghetti code can result, what is harder but infinitely better is to take time to structure the code, make it maintainable, provide a framework

Get a List of People

```
/**
 * Get a list of people from a department in the addressbook
 *
 * @param entrymanager the LDAP Component
 * @param department the organisationalUnit to look in eg. ou=muppets
 * @return a JavaScript Array of JavaScript person Objects
 */
AddressbookManager.getPeople = function(entrymanager, department) {
    var matchAttrs = new HashMap(), people = [];
    department = department ? department : "";
    // prepare the query
    matchAttrs.put(
        "objectClass", this.singleAttribute("person")
    );
    // perform the query
    var results = entrymanager.find(department, matchAttrs);
    // convert to display model
    var it = results.keySet().iterator(), key;
    while (it.hasNext()) {
        key = it.next();
        people[people.length] = this.bindPerson(key + "," + department, results.get(key));
    }
    return people;
}
```

Ross: Here we see a sample of finding with the LDAPEntryManager component. We pass a filter to it, requesting results of type objectClass=person. We specify a department to look in, and return all the people found. We can put some pretty powerful filters in here in the matchAttrs hashmap. Most of this code is our 'framework', designed to make the designers / developers lives easier... very little of it is core to the component.

Manipulating Data

- `var res = em.get("ou=muppets");`
- `var res = em.get("cn=Jeremy Quinn, ou=muppets");`
- `cn=Jeremy Quinn, ou=muppets, ou=addressbook, dc=ldap,`

Ross: Our ldap.base is ou=addressbook,dc=ldap,dc=cocoongt,dc=org, the first get will return info on the muppets department

Ross: Next we grab a contact from the muppets department... creating and updating are all as simple, we can jump to anywhere in the tree by working from right to left

Ross: Our flowscript, using Jeremy's LDAP Cocoon component exploits this, it is just not immediately obvious until you have seen an example... no joins, simple

Empowering

developers and designers to build LDAP Applications

- Javascript skills are on the up (Ajax, Dojo etc)
- The cost of creating LDAP apps is down
- Well thought out frameworks are very productive

Ross: the key – flowscript, javascript skills are on the up, because of the Ajax/Dojo revolution

Ross: javascript is easier, developers are a readily available resource

Ross: We have LDAPEntryManager, all it takes then is to create a nice OO framework, and designers/developers can roll out LDAP web apps rapidly and effectively

○ ● ● Tips and Tricks

for large scale LDAP instances

- come see us after the talk
- look on the Cocoon Wiki to find our talk, sample app and tips