Catacomb
A Database-Backed WebDAV and DASL Repository

Jim Whitehead, Sung Kim
Univ. of California, Santa Cruz
ApacheCon US 2002
Nov 21, 2002

Contents

- WebDAV/DASL Overview
- Catacomb
- Implementation
- Installation/Configuration
- DASL client writing using Neon
- Demo
- Future work/Conclusion

What is WebDAV?

- A protocol for **collaborative authoring** of all document types
  - XML, HTML, word processing, spreadsheets, images
- A Web-based **network file system**
- A **data integration technology** for accessing a wide range of repositories
  - Document mgmt. systems, configuration mgmt. systems, email repositories, filesystems, etc.
- Remote **software engineering** infrastructure
  - Subversion uses DAV/DeltaV
- A **replacement protocol** that can handle email, calendaring, directory lookup and more
  - Could replace: POP, IMAP, CAP, LDAP…

Major WebDAV Clients

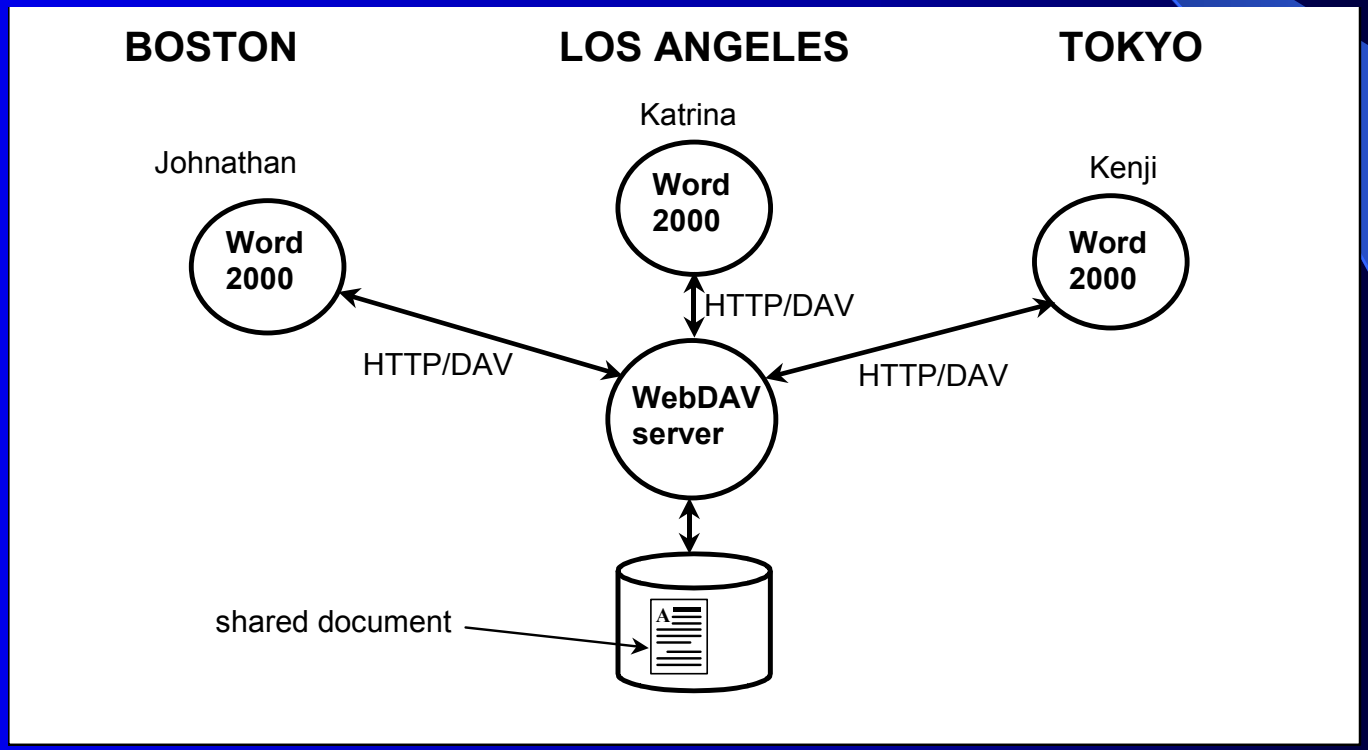
- Application Software:
  - **Microsoft:** Office 2000/XP (Word, Excel, PowerPoint, Publisher)
  - **Adobe:** Photoshop, Illustrator, Acrobat, In Design, FrameMaker
  - OpenOffice (open source)
- Web Site Authoring
  - **Adobe:** Go Live 5/6
  - **Macromedia:** Dreamweaver
- Remote File Access:
  - **Apple:** Mac OS X
  - **Microsoft:** Windows Web Folders, XP Redirector
  - **South River Technologies:** WebDrive
  - **kCura:** kStore Explorer
  - Webdavfs (Linux, open source)
  - Goliath (Mac, open source)
  - Cadaver (Linux/Solaris/Windows, open source)
  - WebDAV Explorer (Java, open source)
- XML editors
  - **Altova:** XML Spy
  - **SoftQuad:** XMetal
  - **Excosoft:** Documentor

Major WebDAV Servers

Apache: mod_dav (over 248,000 sites), Slide
Microsoft: IIS 5/6, Exchange 2000, Sharepoint
FileNet: Panagon ECM
Oracle: Internet File System
Merant: PVCS Dimensions, Content Manager
Xythos: Web File Server
Adobe: Workgroup Server
W3C: Jigsaw
Software AG: Tamino
Hyperwave: Information Server
Novell: Netware 5.1
Sambar: Sambar server
4D: WebSTAR V

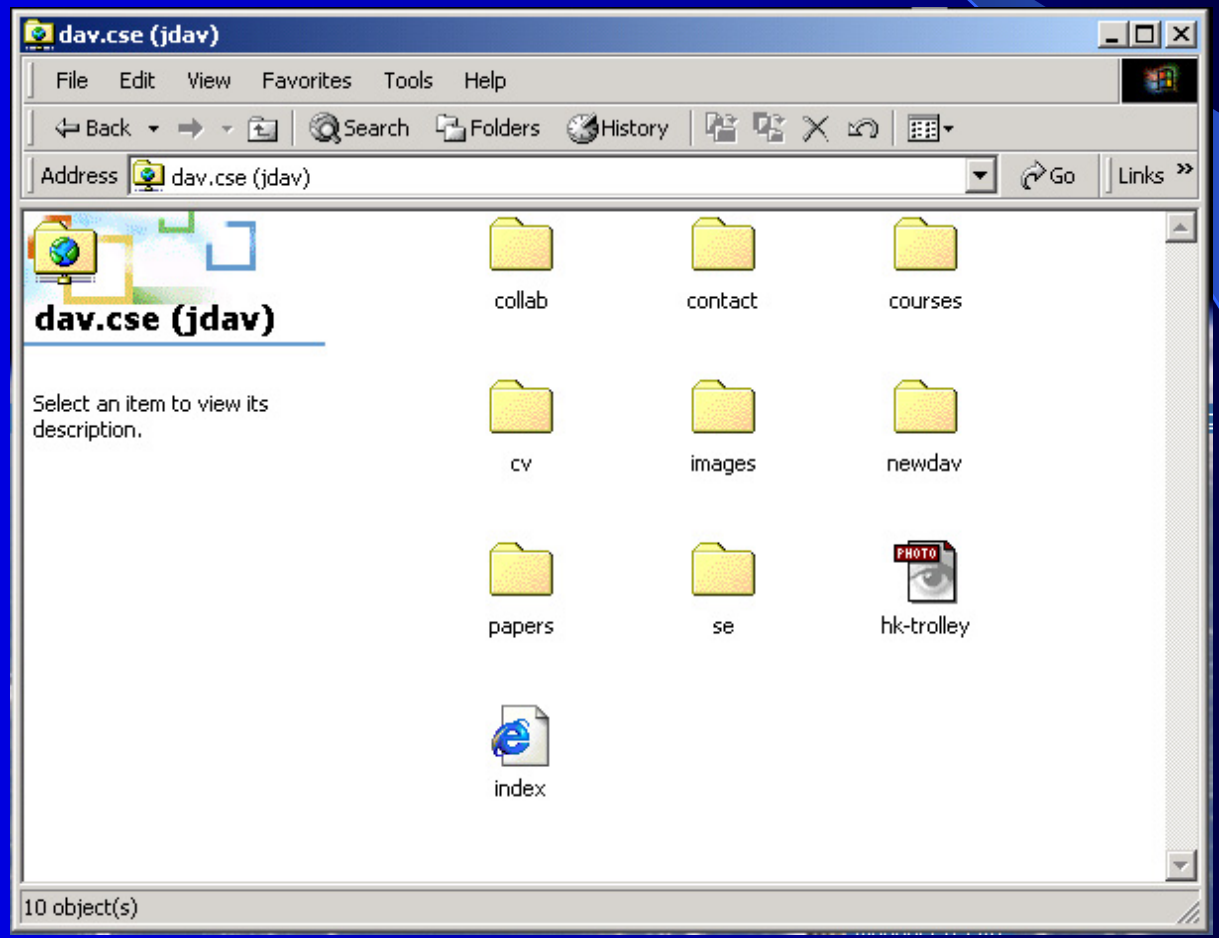
Collaborative Document Authoring

- Three collaborators, in different cities, use Word 2000 to collaborate on a report they are producing together.



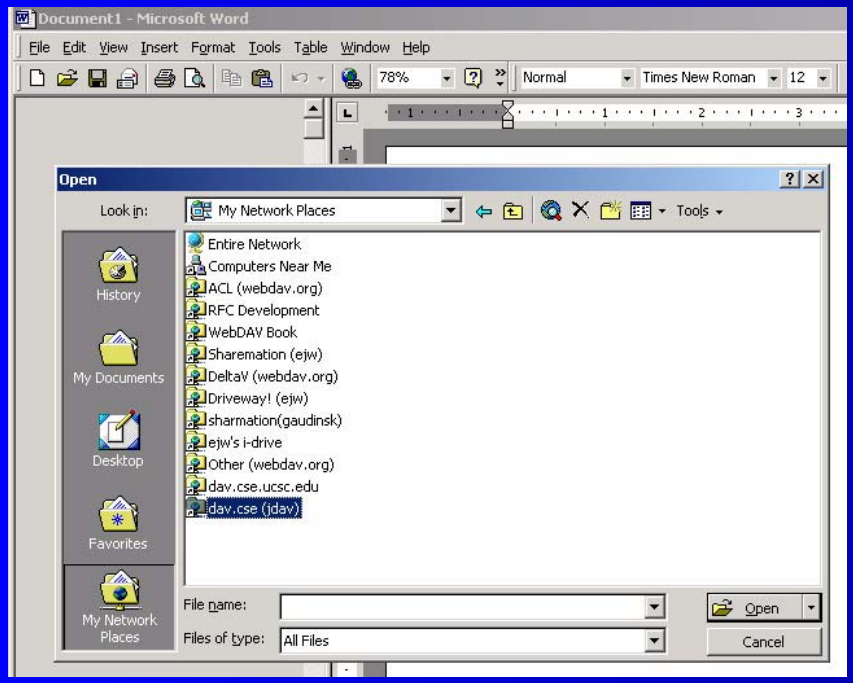
Filesystem View

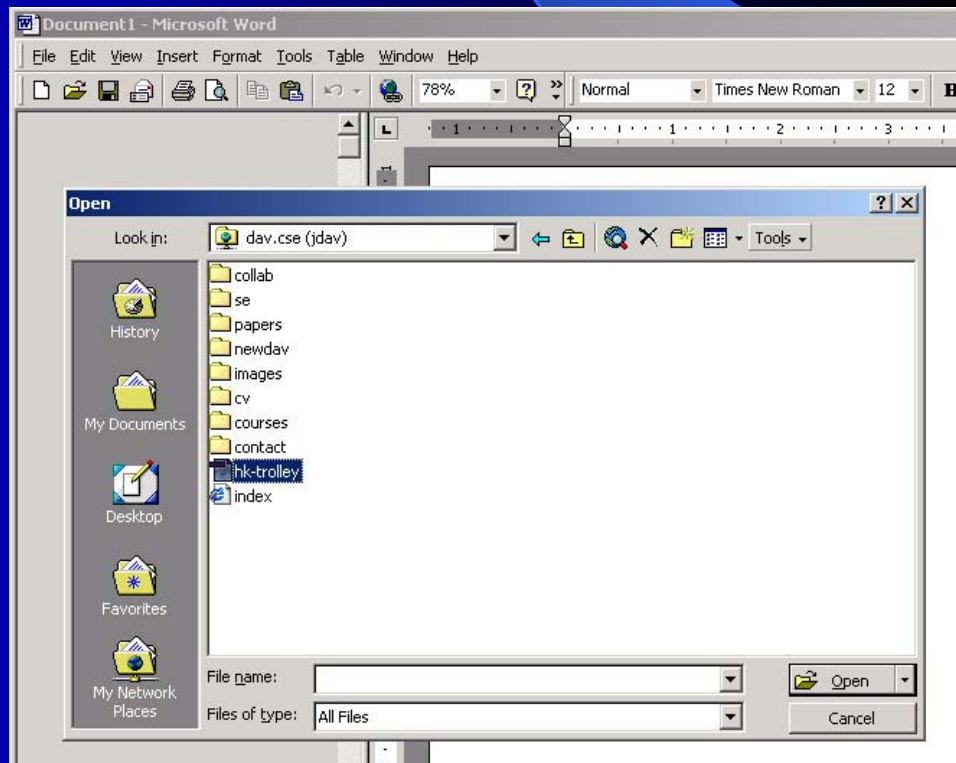
- Exemplars: Web Folders, Mac OS X, WebDrive, TeamDrive, davfs



Document Authoring

- Exemplars: Office 2000/XP: Word, Excel, PowerPoint, as well as XML Spy

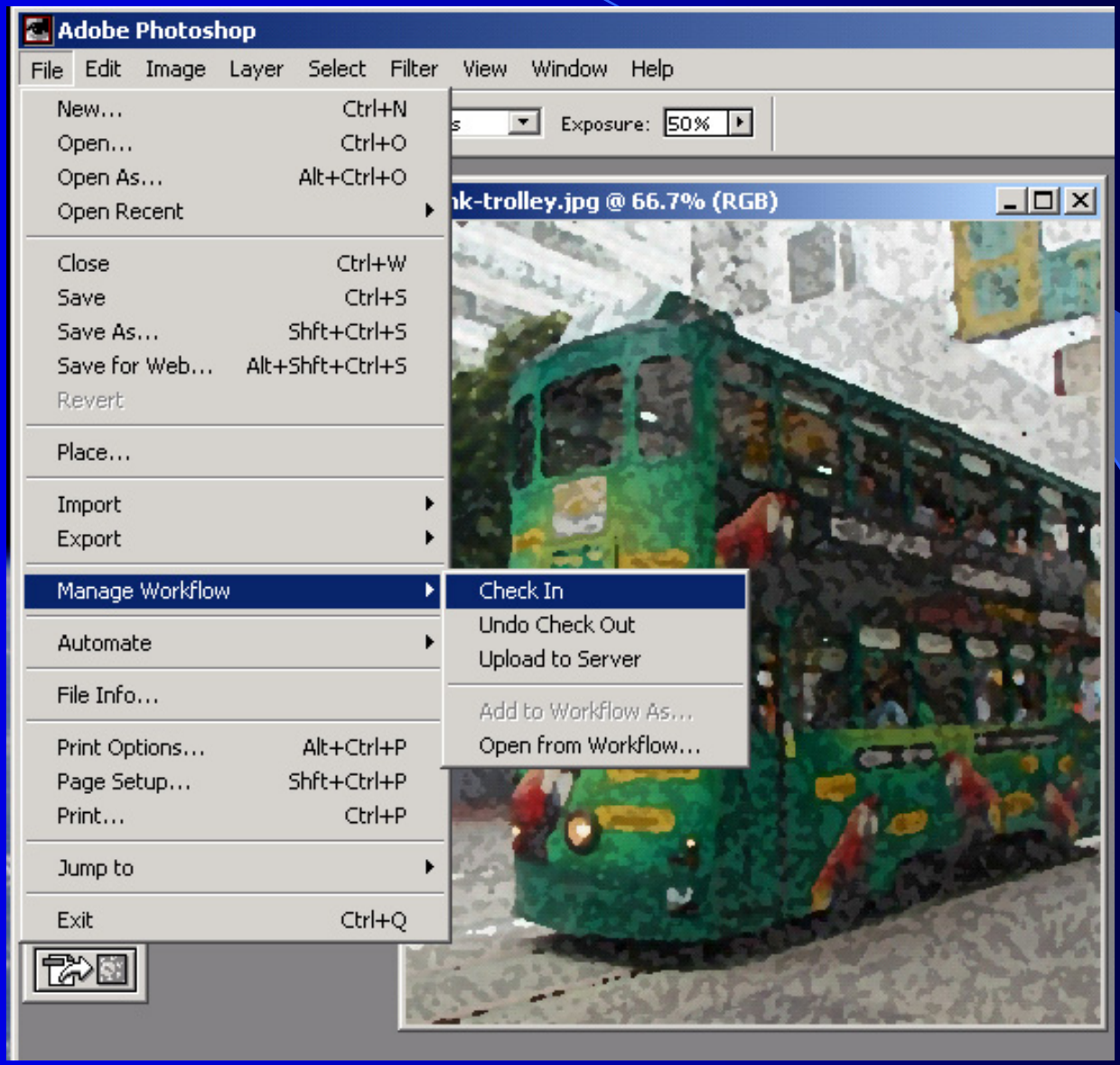




Office: uses filesystem metaphor for WebDAV location

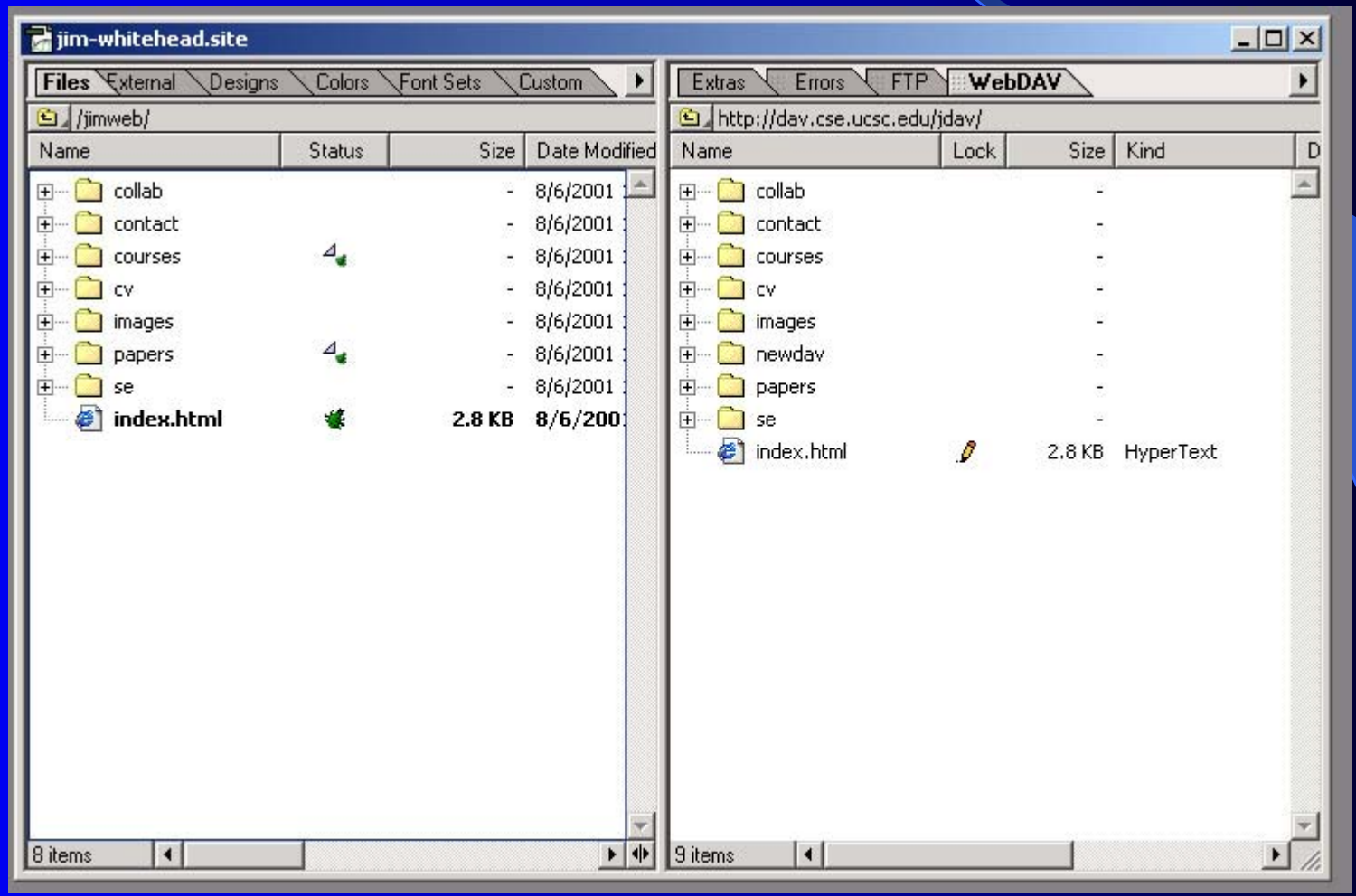
Photoshop

- **Workflow** metaphor for WebDAV location



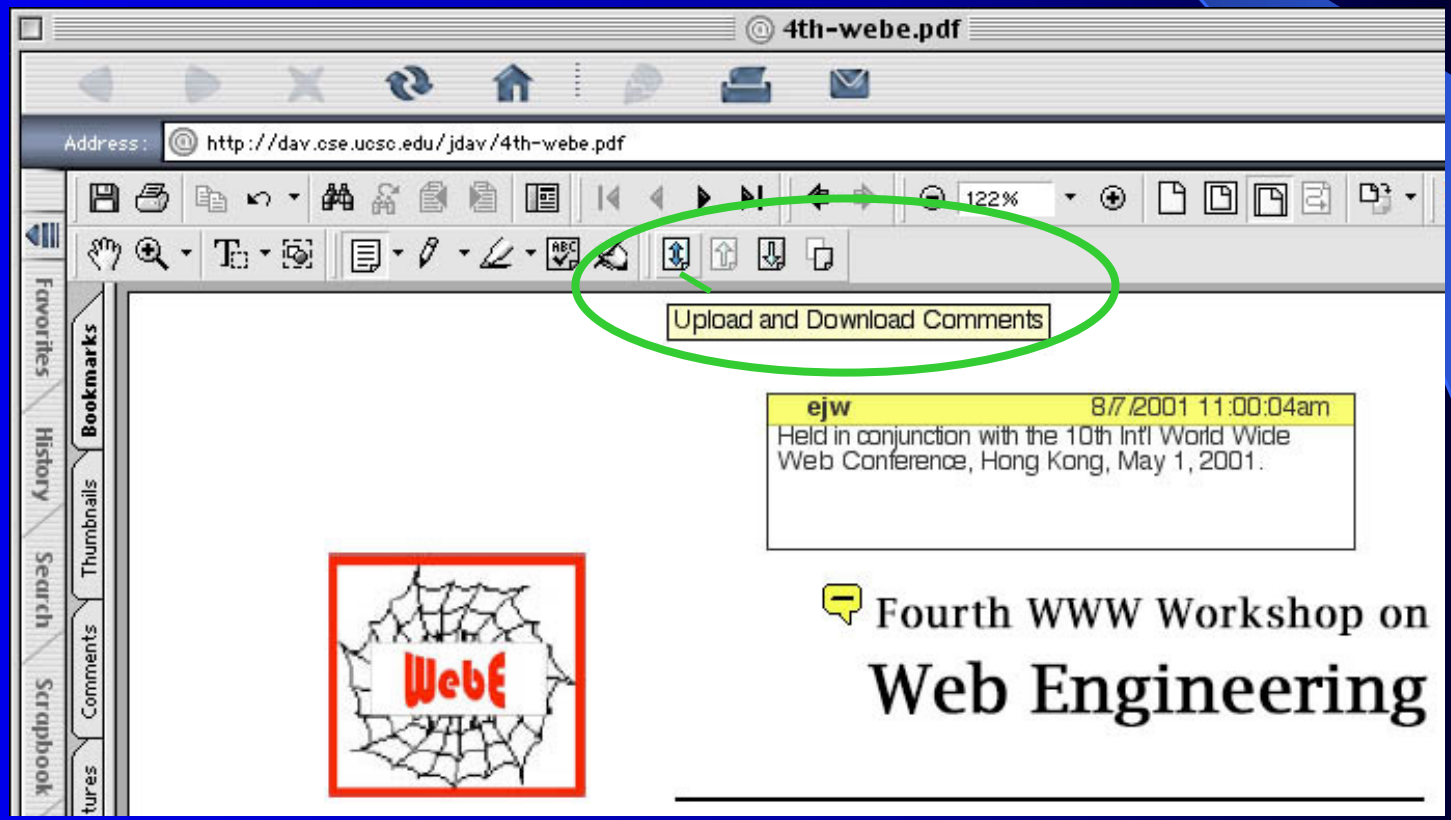
Web Site Authoring

- Exemplars: Go Live 5/6, Dreamweaver
- **Site** metaphor for WebDAV location

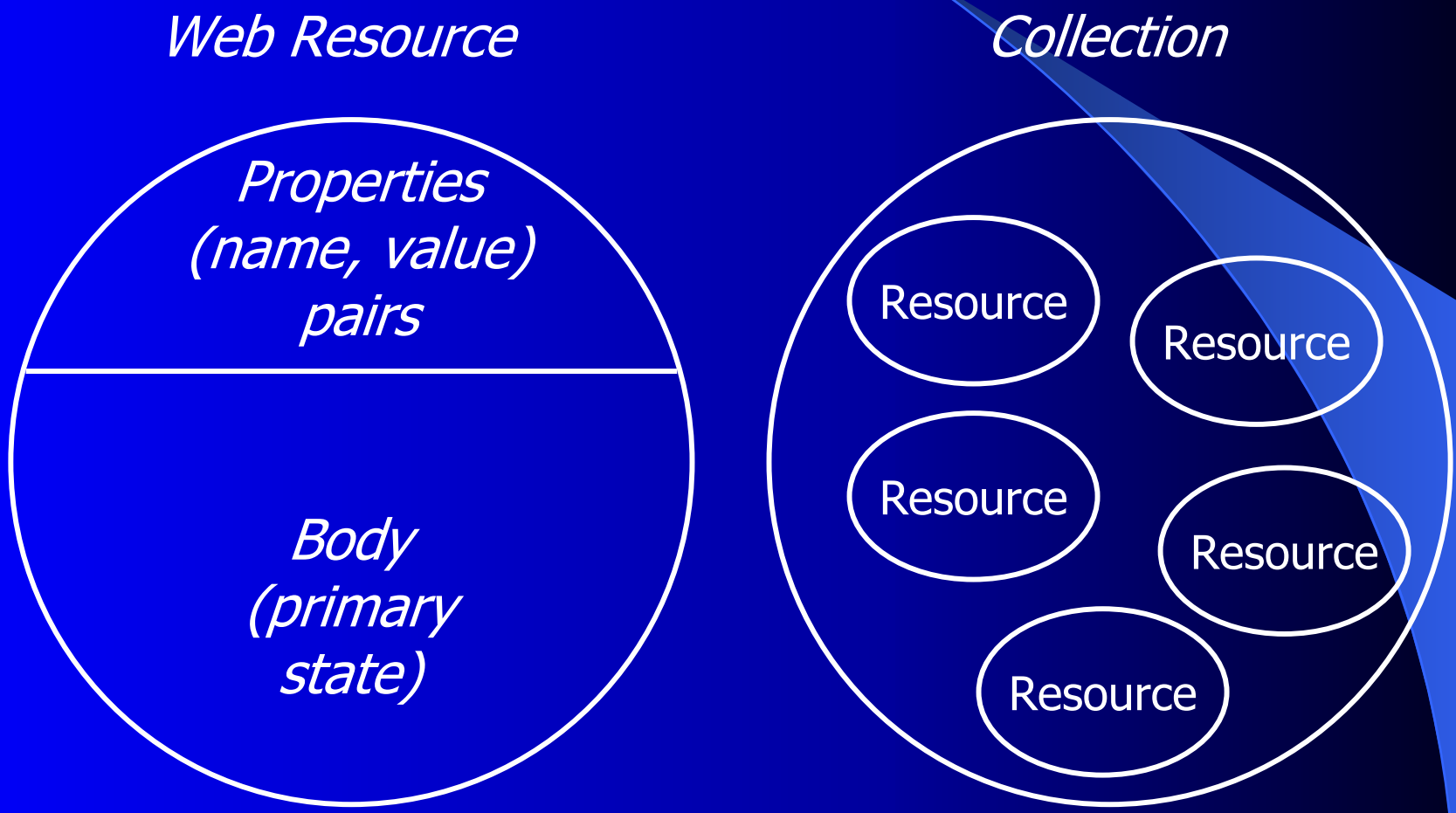


Remote Collaborative Annotation

- Acrobat 5 views a WebDAV location as a storage location for document annotations
  - Annotations are stored in resources separate from the PDF document
    - One collection per document
    - One annotation resource per user (in collection)



WebDAV Data Model



WebDAV Methods

- Resource Management
  - PUT – Creates new resource
  - DELETE – Remove the resource
- Overwrite Prevention
  - LOCK – prevents non-lock holders from writing to the resource
  - UNLOCK – removes a lock
- Metadata Management
  - PROPFIND – read properties from a resource
  - PROPPATCH – write properties on a resource
- Namespace Management
  - COPY – duplicate a resource
  - MOVE – move a resource (preserving identity)
  - MKCOL – create a new collection

Resource Management

- PUT
  - Create a new resource
  - PUT with LOCK
    - LOCK/PUT/UNLOCK
- DELETE
  - Delete a resource
  - Delete a collection

Overwrite Prevention

- LOCK
  - Lock resource
  - Generate Lock-Token
  - Need Lock-Token for UNLOCK and Writing methods
  - Depth 0, 1, infinity
- UNLOCK
  - Unlock resource

Namespace Management

- MKCOL
  - Create a new collection
  - Create Resource Container
- COPY
  - Copy resource
  - Copy collection
- MOVE
  - Move resource/collection

Metadata Management

- PROPPATCH
  - Set properties
  - Dead and Live properties
- PROPFIND
  - Query properties of resource(s)
  - Depth 0, 1, infinity
  - <allprop> or selected properties

DASL: Searching a DAV repository

- The goals of DAV searching and locating – DASL:
  - Server-side search
    - A protocol for accessing server search capabilities
  - Property and content searching
    - Search for properties, content, or combinations of properties and content
  - Multiple scopes
    - Search a collection hierarchy, or just a single resource

DASL Scenario

- Find documents…
  - I have written in the last month
  - Containing key words
  - Written in a specific human language (e.g. French)
  - Having certain property values
- Find XML resources that contain…
  - A specific XML element
  - A specific externally defined DTD
  - A specific XML Namespace

Overview of DASL at Work

- Client constructs a query
  - Uses DAV:basicsearch grammar to construct query
- Client invokes SEARCH method
  - SEARCH is submitted to a **search arbiter** on the server
  - Query is submitted in the request body
- Search arbiter performs the query
- Results returned to client in SEARCH method response

DASL Search

- Client submits a query to a server using SEARCH method
  - Submitted to a search arbiter, which may be different from, or the same as, the search scope
    - For example, to search resources starting at http://svr.com/A/ might need to submit SEARCH to http://svr.com/search-arbiter
  - Query marshalled as XML in the request body using a search grammar
    - DAV:basicsearch grammar must be supported by all
    - Extensible: other search grammars may be used

DASL Query

- **Query = search scope + search criteria + result record definition + sort spec. + search limits**
- **Scope:** the set of resources to be searched
- **Criteria:** an expression against which each resource in the search scope is evaluated (optional)
- **Result:** which properties are returned in a result record
- **Sort spec.:** the ordering of result records in the result set (optional)
- **Limits:** a bound on the number of result records in result set (optional)

DASL Query Example

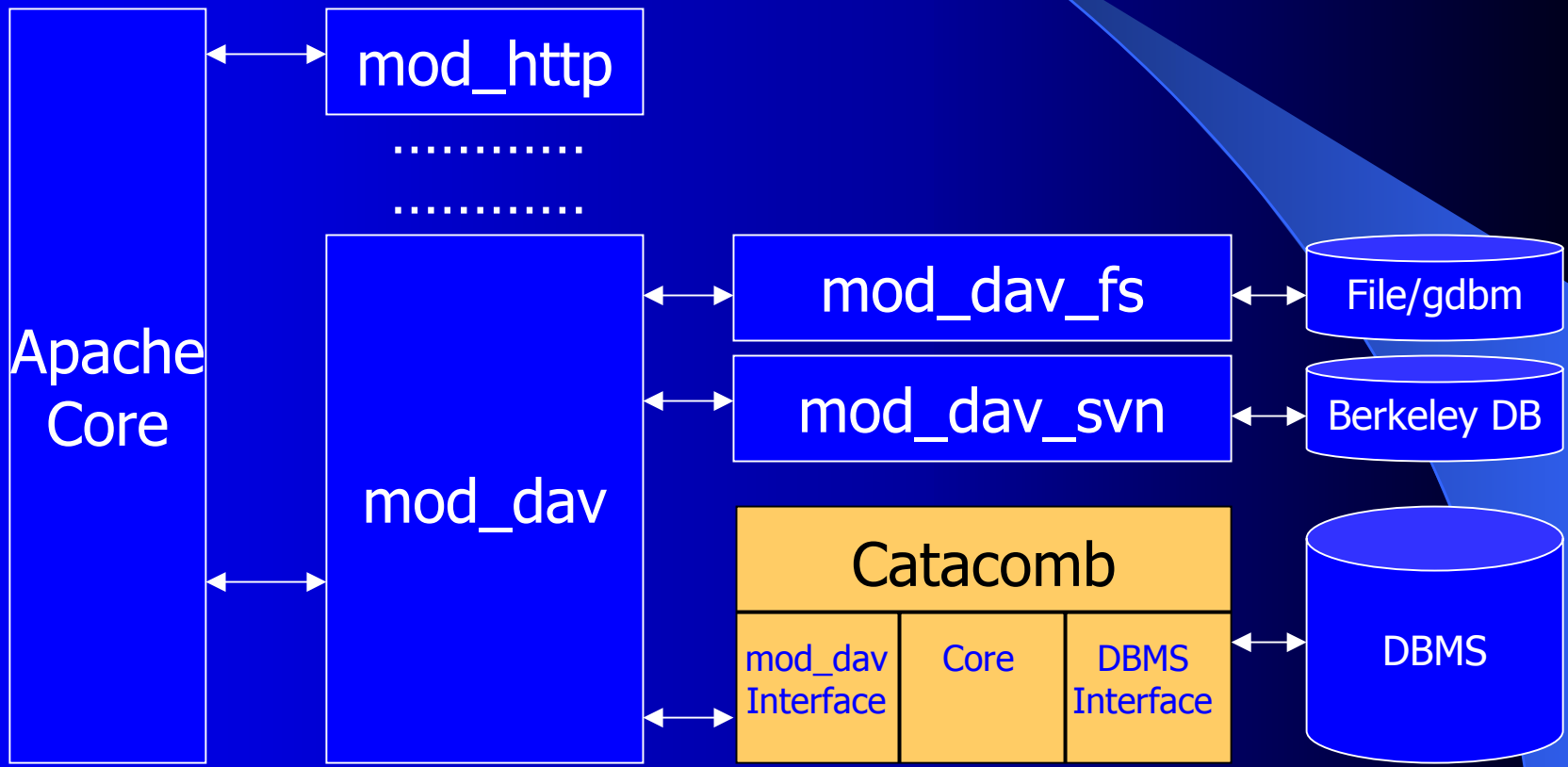
```
<d:searchrequest xmlns:d="DAV:">
  <d:basicsearch>
    <d:select> <d:prop><d:getcontentlength/></d:prop>
    </d:select>
    <d:from>
      <d:scope>
          <d:href>/container1/</d:href> <d:depth>infinity</d:depth>
      </d:scope>
    </d:from>
    <d:where>
      <d:gt> <d:prop><d:getcontentlength/></d:prop>
      <d:literal>10000</d:literal></d:gt>
    </d:where>
    <d:orderby>
      <d:order> <d:prop><d:getcontentlength/></d:prop>
   <d:ascending/> </d:order>
    </d:orderby>
  </d:basicsearch>
</d:searchrequest>
```

Catacomb

Catacomb Overview

- WebDAV repository module for mod_dav
- DAV 1,2 and DASL implementation
- Search capability
- Easy resource management using DBMS
  - Contents, properties, lock information
  - Facilitates implementation of DeltaV, Bindings
- First open source implementation of DASL

mod_dav/Catacomb Architecture



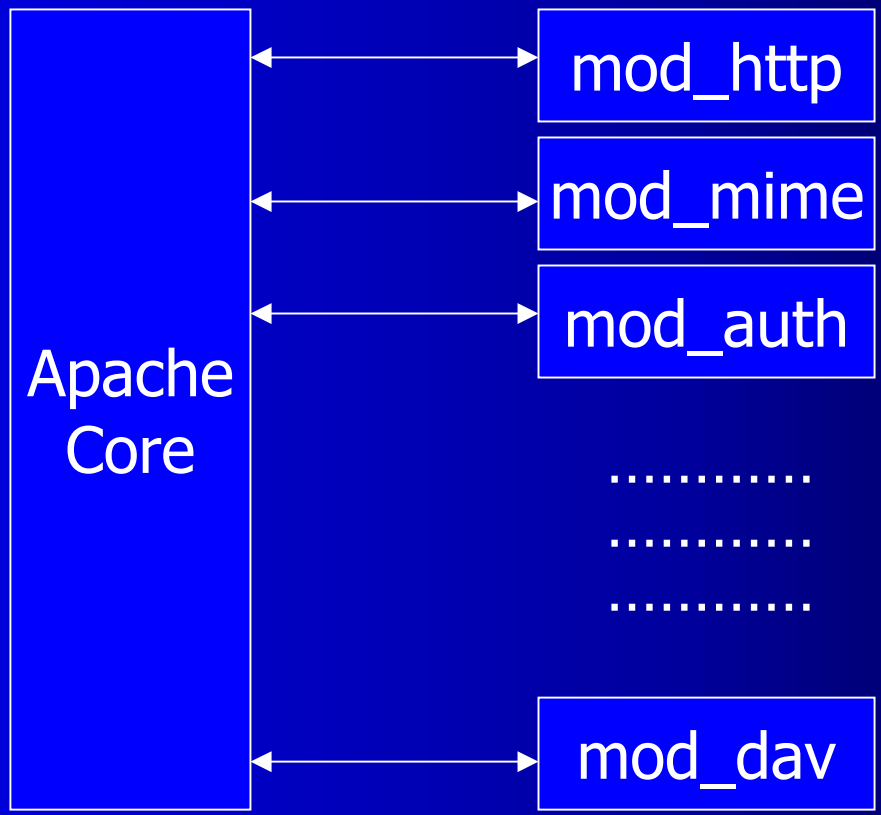
Catacomb vs mod_dav_fs

- Why not use mod_dav_fs?
  - Devil is in the details:
    - mod_dav_fs uses gdbm to save properties
    - mod_dav_fs creates one gdbm file per resource
  - Consequence:
    - A single DASL query needs to open many files
    - Implementation of complex queries is difficult
    - Full text search is expensive
  - Need a SQL processor

Catacomb & DBMS

- Why DBMS?
  - Facilitates management of data/metadata and containment relations
  - Supports SQL-based searching
  - Can support binary searching
    - Save text content and binary content at the same time
    - PDF file stored as binary, but abstract stored as text
  - Full text searching
  - Not a hierarchical structure
    - Only URIs represent the hierarchy
    - Supports referential containment
  - Fast “depth infinity” operations

Apache2 Architecture



Catacomb Implementation

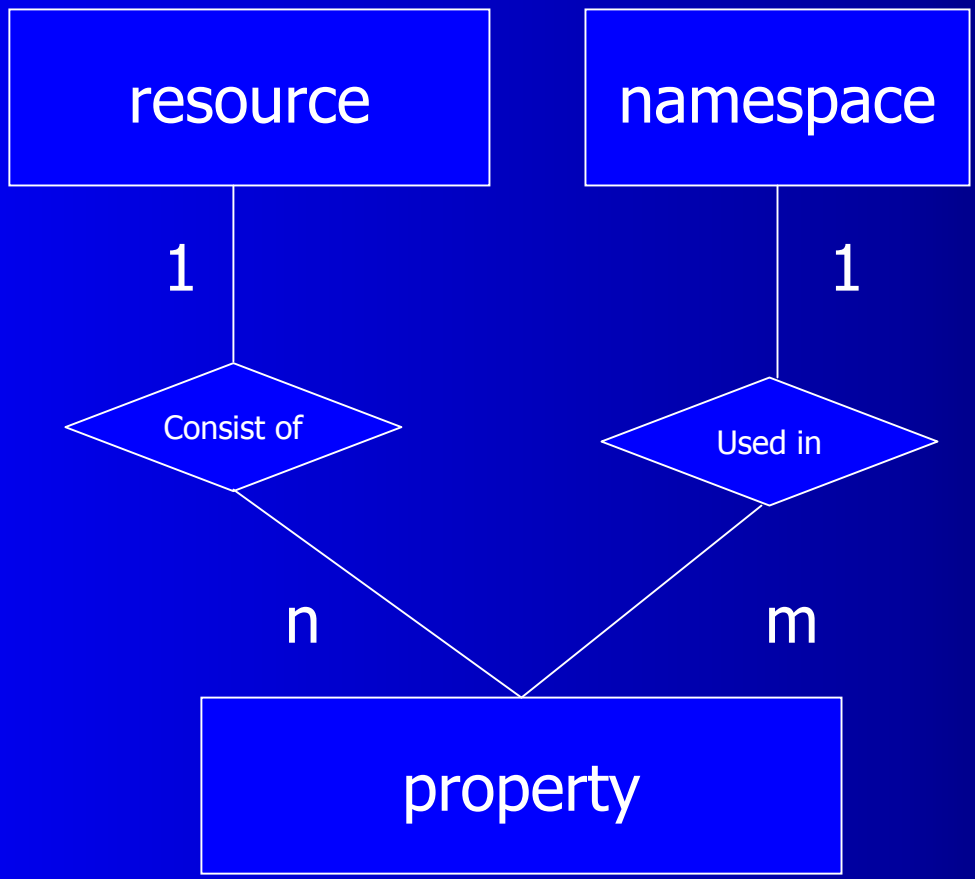
mod_dav Hook

```
typedef struct {
    const dav_hooks_repository *repos;
    const dav_hooks_propdb *propdb;
    const dav_hooks_locks *locks;
    const dav_hooks_vsn *vsn;
    const dav_hooks_binding *binding;
    const dav_hooks_search *search;
    void *ctx;
} dav_provider;
```

mod_dav Repository Hook

```
/* Repository provider hooks */
struct dav_hooks_repository
{
    ...
    dav_error * (*create_collection)(
        dav_resource *resource
    );
    ...
}
```

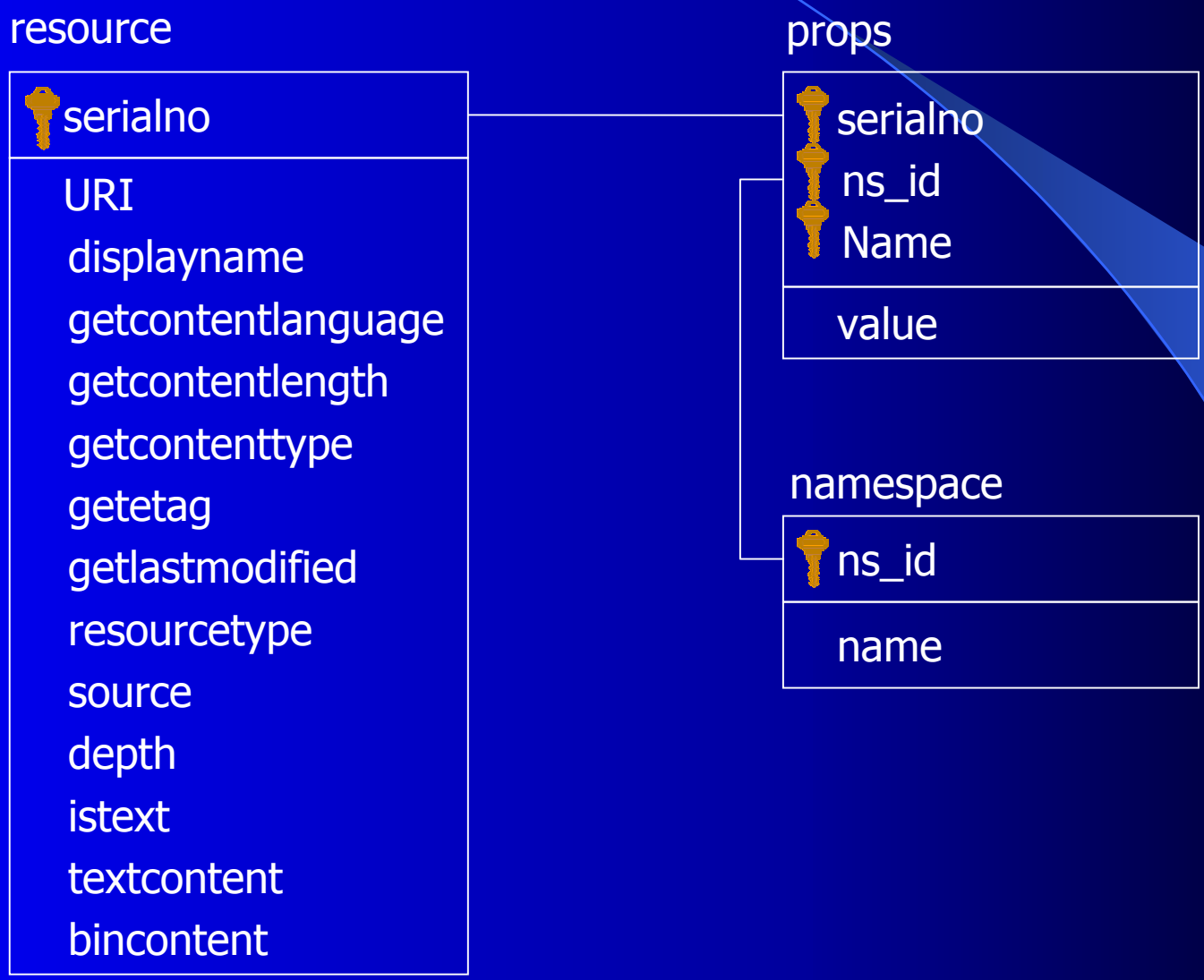
Database Tables



lock

locknull

Resource Schema



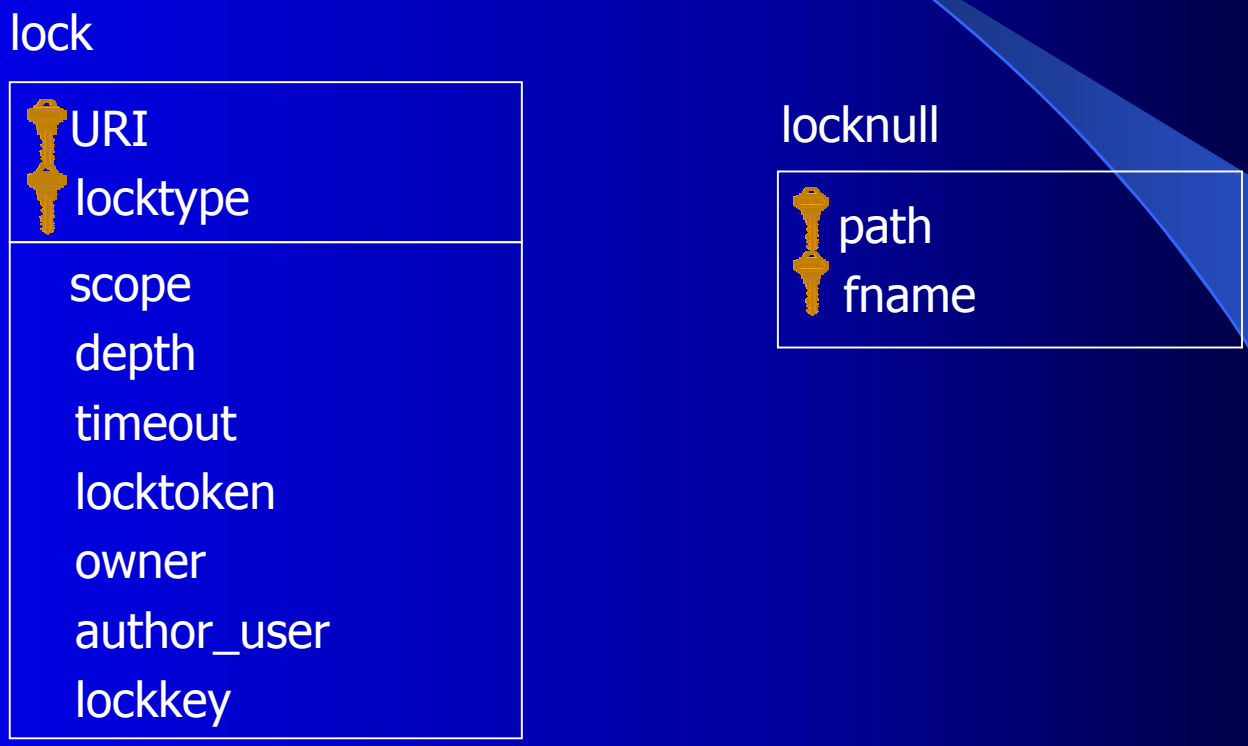
Properties Schema

- Live properties are stored in ‘resource’ table
- Dead properties are stored in ‘property’ table
- Live properties are fixed
- Dead property name is not fixed
- Needs complicated SQL to deal with dead property

PROPFIND

- Depth infinity needs only one SQL
  - Select * from resource where URL like '/repos/%'
- Dead props need one SQL per resource
- Better than mod_dav_fs
  - Opens and stats each resource recursively
  - Opens each resource's dbm file to find properties

Lock Schema


lock
URI
locktype
scope
depth
timeout
locktoken
owner
author_user
lockkey
locknull
path
fname

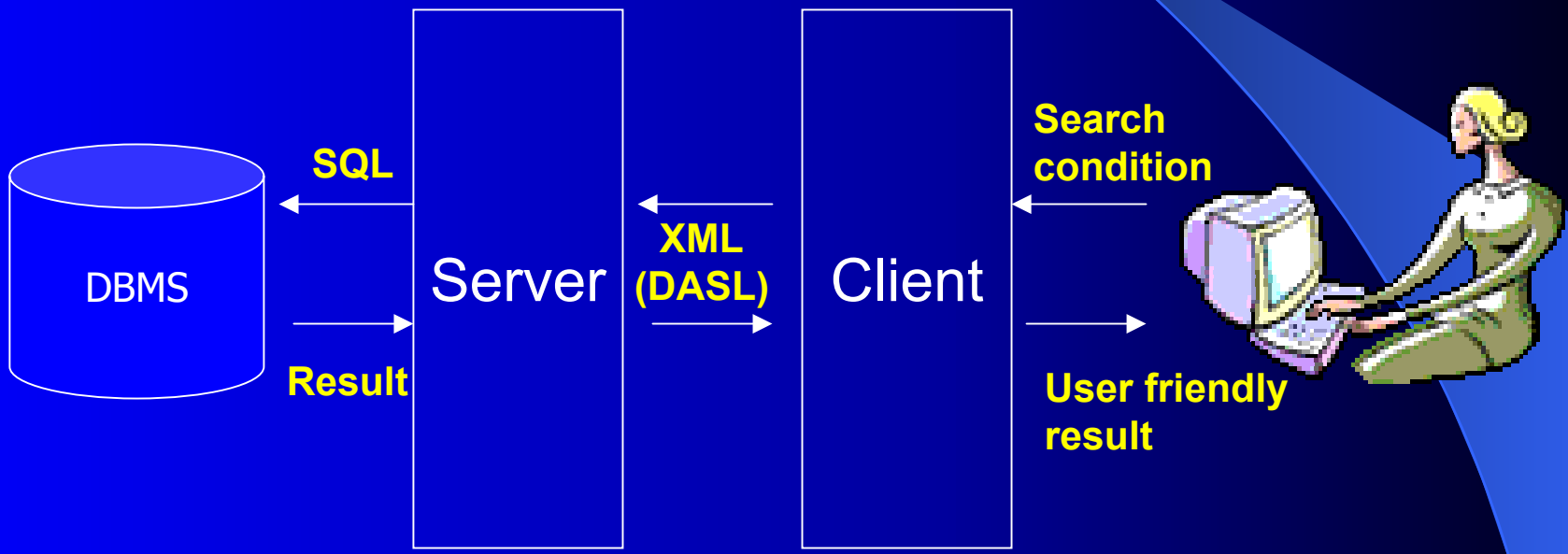
Lock Schema

- URI is key for lock
- Lock token
- Lock owner
- Lock timeout
- Null Lock path, filename

LOCK/UNLOCK

- URI is key for LOCK/UNLOCK
- LOCK
  - Add lock record in DBMS
  - Check DBMS for any writing action
- UNLOCK
  - Remove record in DBMS

SEARCH Overview



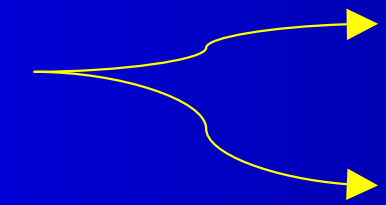
SEARCH Query Parser

```
<d:searchrequest xmlns:d="DAV:">
 <d:basicsearch>
  <d:select>
   <d:prop>
    <d:displayname/>
    <d:foo/>
    <d:bar/>
   </d:prop>
  </d:select>
  <d:from>
   <d:scope>
    <d:href>/dbms</d:href>
    <d:depth>infinity</d:depth>
   </d:scope>
  </d:from>
  <d:where>
   <d:gt>
    <d:prop><d:bar/></d:prop>
    <d:literal>2518</d:literal>
   </d:gt>
  </d:where>
 </d:basicsearch>
</d:searchrequest>
```

```
SELECT
        dasl_resource.displayname,
        t.name, t.value
FROM
        dasl_resource
        LEFT JOIN
        dasl_property t USING (serialno)
        LEFT JOIN
        dasl_property bar_t USING (serialno)
WHERE
        ( bar_t.name = 'bar' AND
        bar_t.value > 2518 )
AND
        ( t.name = 'foo' OR t.name = 'bar' )
```

SEARCH Query Parser

```
<d:searchrequest xmlns:d="DAV:">
 <d:basicsearch>
  <d:select>
   <d:prop>
    <d:displayname/>
    <d:foo/>
    <d:bar/>
   </d:prop>
  </d:select>
  <d:from>
   <d:scope>
    <d:href>/dbms</d:href>
    <d:depth>infinity</d:depth>
   </d:scope>
  </d:from>
  <d:where>
   <d:gt>
    <d:prop><d:bar/></d:prop>
    <d:literal>2518</d:literal>
   </d:gt>
  </d:where>
 </d:basicsearch>
</d:searchrequest>
```



```
SELECT
        dasl_resource.displayname,
        t.name, t.value
FROM
        dasl_resource
        LEFT JOIN
        dasl_property t USING (serialno)
        LEFT JOIN
        dasl_property bar_t USING (serialno)
WHERE
        ( bar_t.name = 'bar' AND
        bar_t.value > 2518 )
AND
        ( t.name = 'foo' OR t.name = 'bar' )
```

SEARCH Query Parser

```
<d:searchrequest xmlns:d="DAV:">
 <d:basicsearch>
  <d:select>
   <d:prop>
    <d:displayname/>
    <d:foo/>
    <d:bar/>
   </d:prop>
  </d:select>
  <d:from>
   <d:scope>
    <d:href>/dbms</d:href>
    <d:depth>infinity</d:depth>
   </d:scope>
  </d:from>
  <d:where>
   <d:gt>
    <d:prop><d:bar/></d:prop>
    <d:literal>2518</d:literal>
   </d:gt>
  </d:where>
 </d:basicsearch>
</d:searchrequest>
```

```
SELECT
        dasl_resource.displayname,
        t.name, t.value
FROM
        dasl_resource
        LEFT JOIN
        dasl_property t USING (serialno)
        LEFT JOIN
        dasl_property bar_t USING (serialno)
WHERE
        ( bar_t.name = 'bar' AND
        bar_t.value > 2518 )
AND
        ( t.name = 'foo' OR t.name = 'bar' )
```

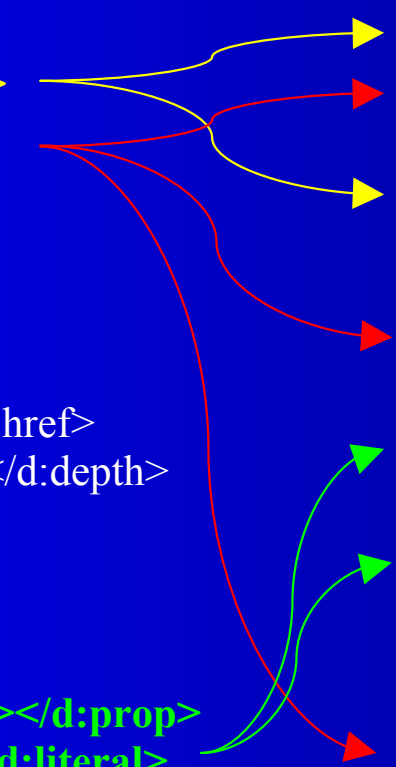
SEARCH Query Parser

```
<d:searchrequest xmlns:d="DAV:">
 <d:basicsearch>
  <d:select>
   <d:prop>
    <d:displayname/>
    <d:foo/>
    <d:bar/>
   </d:prop>
  </d:select>
  <d:from>
   <d:scope>
    <d:href>/dbms</d:href>
    <d:depth>infinity</d:depth>
   </d:scope>
  </d:from>
  <d:where>
   <d:gt>
    <d:prop><d:bar/></d:prop>
    <d:literal>2518</d:literal>
   </d:gt>
  </d:where>
 </d:basicsearch>
</d:searchrequest>
```

```
SELECT
        dasl_resource.displayname,
        t.name, t.value
FROM
        dasl_resource
        LEFT JOIN
        dasl_property t USING (serialno)
        LEFT JOIN
        dasl_property bar_t USING (serialno)
WHERE
        ( bar_t.name = 'bar' AND
        bar_t.value > 2518 )
AND
        ( t.name = 'foo' OR t.name = 'bar' )
```

SEARCH Query Parser

```
<d:searchrequest xmlns:d="DAV:">
 <d:basicsearch>
  <d:select>
   <d:prop>
    <d:displayname/>
    <d:foo/>
    <d:bar/>
   </d:prop>
  </d:select>
  <d:from>
   <d:scope>
    <d:href>/dbms</d:href>
    <d:depth>infinity</d:depth>
   </d:scope>
  </d:from>
  <d:where>
   <d:gt>
    <d:prop><d:bar/></d:prop>
    <d:literal>2518</d:literal>
   </d:gt>
  </d:where>
 </d:basicsearch>
</d:searchrequest>
```



```
SELECT
        dasl_resource.displayname,
        t.name, t.value
FROM
        dasl_resource
        LEFT JOIN
        dasl_property t USING (serialno)
        LEFT JOIN
        dasl_property bar_t USING (serialno)
WHERE
        ( bar_t.name = 'bar' AND
        bar_t.value > 2518 )
AND
        ( t.name = 'foo' OR t.name = 'bar' )
```

Installation

Installation-Apache

- Apache 2.0.42 or later
- Compile apache2 with mod_dav
  - `./configure –enable-dav`
  - `make; make install`

Installation-MySQL

- MySQL 3.22 or later
- File size limitation
  - MySQL 3 : Up to 16M
  - MySQL 4 : Up to 2G
- Set option with safe_mysqld
- Or edit startup script
  - `--set-variable=max_allowed_packet=16M`

Installation-Catacomb

- Download catacomb tar ball
  - http://www.webdav.org/catacomb
- Configure with apache2 and MySQL dir
  - `./configure`
    `–with-apache=/usr/local/apache2`
    `–with-mysql=/usr/local`
- Build
  - `make; make install`

Installation-DB Tables

- Create Database
  - `mysqladmin create repos`
- Create Tables
  - `mysql repos < table.sql`
- Import initial data
  - `mysql repos < data.sql`

Configuration-Apache

- Apache2 per server configure – DB

```
DavDBMSHost localhost
DavDBMSDbName repos
DavDBMSId   myid
DavDBMSPass “mypass”
DavDBMSTmpDir /tmp/
```

- Apache2 per directory configure – Location

```
<location /repos>
   Dav repos
   ModMimeUsePathInfo on
</Location>
```

Configuration-Start Apache

- Apache Start
  - apachectl start
- Testing Catacomb Server

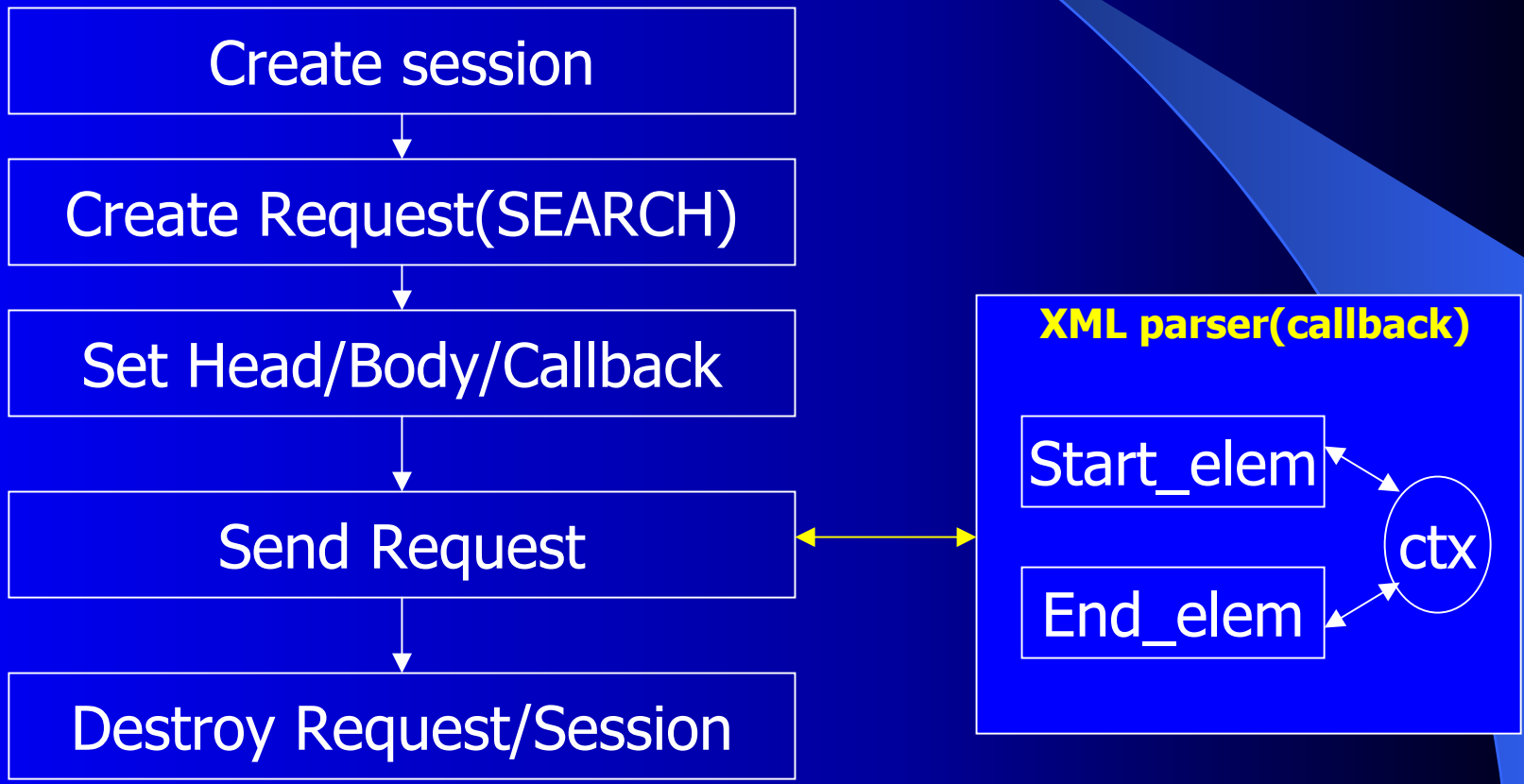
```
ocean 5> telnet ocean 80
Trying 128.114.51.104...
Connected to ocean.
OPTIONS /repos HTTP/1.1
Host: ocean

HTTP/1.1 200 OK
Date: Sat, 21 Sep 2002 00:33:06 GMT
Server: Apache/2.0.41-dev (Unix) DAV/2 SOAP/1.1 Catacomb/0.7.4
DAV: 1,2
DAV: <http://apache.org/dav/propset/fs/1>
MS-Author-Via: DAV
Allow: OPTIONS,GET,HEAD,POST,DELETE,TRACE,PROPFIND,
       PROPPATCH,COPY,MOVE,LOCK,UNLOCK,SEARCH
DASL: <DAV:basicsearch>
Content-Length: 0
Content-Type: text/plain; charset=ISO-8859-1
```

Client Writing Using Neon

Neon Overview

- HTTP/DAV client library
  - C language
  - PERL wrapper
    - ftp://ftp.dev.ecos.de/pub/perl/webdav/HTTP-Webdav-0.1.18-0.17.1.tar.gz
  - Developed by Joe Orton
- Features:
  - Easy to extend with new methods
  - Supports SSL and Proxies
  - Supports Basic and Digest authentication
- http://www.webdav.org/neon

Neon Processing Sequence



Neon Sample Code (1)

```
/* Create Session
  Creates a 'session' struct variable */
 sess = ne_session_create(scheme, host, port);

/* Create Method
 Creates a 'session' struct variable */
req = ne_request_create(sess, "SEARCH", uri);

/* Set user Head*/
 ne_add_request_header(req, "Content-Type",
                                          NE_XML_MEDIA_TYPE);
 ne_add_depth_header(req, depth);

 /* Set Body */
 char *data = "<?xml version=\"1.0\"?> …." ;
 ne_set_request_body_buffer(req, data,  strlen(data));
```

Neon Sample Code (2)

```
/* Set Callback, XML Parser
    start_element : call back function for open element
    end_element   : call back function for closing element  */
search_parser = ne_xml_create();
ne_xml_push_handler(search_parser, search_elements,
                                    validate_search_elements,
                                   start_element, end_element, sctx);
ne_add_response_body_reader(req, search_accepter,
                                          ne_xml_parse_v, search_parser);

/* Send Request. Network connection */
ret = ne_request_dispatch(req);
    ...
/* Destroy request and session */
 ne_request_destroy(req); ne_session_destroy(session);
```

Demo

- Catacomb server
- Neon/Cadaver_DASL
- SEARCH method actually sent

Future Work

- Database abstraction layer – support multiple DBMS
- Improve SEARCH function
- Implement WebDAV family protocols
  - Delta-V – Version Control
    - Work in process
  - ACL – Access control
  - WebDAV Binding – referential containment

Conclusion

- Catacomb is good for:
  - Digital library
  - Documentation management
  - Content management
  - Collaborated web authoring
  - With Search capability
- Catacomb is an open source project
  - We welcome contributors
  - http://www.webdav.org/catacomb

Questions?

http://webdav.org/catacomb

catacomb@webdav.org