WebDAV and Apache

Greg Stein
gstein@lyra.org, http://www.lyra.org/

Abstract

WebDAV is an exciting new technology for the World Wide Web. WebDAV stands for Web-Based Distributed Authoring and Versioning, and provides a way to remotely author and manage your Web servers (whether you are an author or an administrator). The WebDAV protocol is specified by RFC 2518 and is now available as an add-on module for the Apache Web server. This paper presents an overview of WebDAV, its benefits for users, and scenarios for effective deployment. It also details how to set up the mod_dav Apache module, and describes the available tools and applications to use with your new WebDAV-enabled server. In closing, this paper discusses the future directions of the WebDAV protoco.

WebDAV Overview

WebDAV stands for "Web-based Distributed Authoring and Versioning." Simply speaking, it provides a mechanism for authors to create documents on a web server and to manage their organization. WebDAV, or "DAV" for short, is driven by a desire to create interoperable tools for distributed web

authoring. Where the World Wide Web has generally been a read-only environment, DAV seeks to turn it into a *writeable* medium.

DAV is specified by RFC 2518[1] as a set of extensions to the HTTP protocol. Just like HTTP, DAV operates on all kinds of document types, such HTML pages, images, text files, or PDF documents.

It is important to note that RFC 2518 does not specify any versioning mechanisms. Versioning is a very large problem set, so it was deferred in order to get a basic, core level of WebDAV completed and available for use.

Benefits of Using DAV

Users, authors, and server administrators can receive benefits from the deployment of DAV technology. There are also technical benefits for developers, security personnel, and network administrators.

Users, in this context, refer to people who are visiting a web site. DAV can assist these users with providing additional metadata about the documents they are viewing. For example, information about the author, the revision history, or the last modification can be fetched from the DAV server and made available to the user. This can be performed in a consistent fashion rather than using ad hoc techniques, such as placing the last-modification date at the bottom of a web page. These ad hoc techniques may not even be in use by the site authors, may be used inconsistently throughout the site, and cannot easily be interpreted and used by sophisticated user agents. Historically, authors have provided users with an `index.html` document to provide an index into the contents of a directory or site. Using DAV, a user can fetch complete listings of a server's contents, subject to access restrictions. The user's browser could display this listing in varying formats according to the user's preferences, rather than according to the author's desires. In fact, the Web Folders add-on to Internet Explorer 5 does exactly this – it displays the listings in the same fashion as other resources on the user's local disk. Web Folders provides icons, varying layouts, and property pages to view detailed information.

For the author, DAV finally provides a way to consistently move documents from their client, authoring machine to the web server. The author no longer needs to invoke FTP and then remember how the filesystem presented by FTP maps to the URL namespace provided by the web server. An author can specify the exact URL where the document should be stored. In addition, DAV provides mechanisms to manage existing content on the server – moves, copies, and deletes are possible. The author can also attach metadata to the documents – possibly for their own use, for browsing by users, or for tracking among the authoring team members. A DAV server also provides overwrite protection for multiple authors to ensure they do not destroy changes made by others when a (changed) document is placed on the server.

The server administrator can use DAV to increase the flexibility in the mapping of the URL namespace to an underlying filesystem or database. Portions may be placed into a database, others into a high-performance filesystem, and still others onto a network data repository. All interactions will come through the web server, which can select the appropriate repository. Historically, web documents have had to reside in the filesystem so the authors can easily access them with editing tools. A second benefit is that the server administrator is not even required to provide "system" accounts for the authors – authors' authentication will happen through the web server. By preventing access, via system accounts, to the underlying system, the administrator can greatly enhance the reliability and security of the server.

From a technical standpoint, DAV introduces a number of features to the Web. These can be used by developers for new types or enhanced features in their products. These new features are discussed in the next section.

The fact that DAV builds upon HTTP can simplify a network or security administrator's job – there is a single protocol (HTTP) to worry about. Separate tunnels, proxies, and policies are no longer required. In fact, it becomes possible to throw out plain-text password mechanisms, such as the one used by the standard FTP protocol, and replace them with stronger system. Certificates, Kerberos, or tunneling passwords through encrypted channels, such as SSL, all becomes possible with HTTP.

DAV's Features

WebDAV introduces several key, new features for developers to use: properties, overwrite protection, namespace management, new infrastructure and reuse of existing infrastructure, and a basis for replacing other protocols.

Note: some new terminology will be used below – please see the Appendix for a quick discussion of terminology used by DAV

.

Properties

Properties, also known as "metadata", are stored as name/value pairs on the server. Every resource, whether a member resource or a collection resource, may have properties associated with it. Some properties will be defined by the server and are called "live properties". Other properties are client-defined and are called "dead properties".

A property name is a URI, which ensures that it is universally unique. Example property names are "DAV:getlastmodified" or "http://apache.org/dav/props/executable". A property value is a well-formed XML fragment.

Properties can store items such as author, title, modification time, or size. If a property is server-defined ("live"), then it may be read-only and/or the (XML) contents may be required to fit a particular pattern. Dead properties' formatting and integrity are specified and managed by the clients that store them on the server.

Overwrite protection

WebDAV provides different types of locks that clients may use for arbitrating access to resources. These locks can be applied to individual resources, or an entire "tree" of resources. A lock has different pieces of state, including a timeout, an owner, a "lock token", and an associated, authenticated user.

It is important to note that timeouts are associated with the locks. A client may request a particular timeout, and the server may choose to honor or ignore that value. In addition, the client must also assume that a lock may disappear at any time – typically, for administrative reasons, but it could also be caused by, say, a server crash. These features of the locks may seem to create too much indeterminism for a DAV client, but the client actually has a number of ways to cope with the situation. For example, the client can store the resource's "entity-tag" when the lock is acquired. If the lock disappears, then the client can reacquire the lock and compare it saved entity-tag against the resource's current entity-tag. If no change has occurred in the entity-tag, then the client knows it will not overwrite any changes when the document is saved. If a change *has* occurred, then the client can work out a solution with the user.

Namespace management

In this context, "namespace" refers to the URL namespace exposed by a web server. DAV provides mechanisms to manage this namespace through copies, moves, deletes, and construction of resources.

Infrastructure: old and new

DAV is built using HTTP, so it inherits all of the benefits of the existing HTTP infrastructure. Specifically, items such as strong authentication, encryption, proxy and firewall navigation, and caching become available to DAV. The worldwide deployment of HTTP helps greatly, too, in providing a base for DAV to be used worldwide. There are also more subtle benefits that HTTP provides: a very large talent pool of people who understand HTTP, a large number of books, tools, and applications that deal with HTTP, and a "critical mass" for further development and deployment.

Building on top of this infrastructure, DAV brings its new features: writeable resources, properties, overwrite protection, and namespace management. As additional DAV capabilities are introduced

(such as access control and versioning), these will continue to grow the infrastructure. This expansion of the existing infrastructure can then lead to more features and capabilities.

A replacement protocol

There are a number of proprietary, specialized, or alternative mechanisms for transferring documents to a web server. DAV can easily replace these protocols and, at the same time, provide even more functionality.

For example, using FTP to move files to a web server is problematic in a number of areas: plain-text passwords, needing to map a filesystem view to a URL namespace, and inconsistent directory listings. As an alternative, DAV can use stronger authentication ("digest", Kerberos, or even certificates), it can encrypt the channel, there is no mapping between namespaces, and DAV's form of a directory listing can reliably return modification dates, file size, names, and file types.

Proprietary protocols, such as those used by Microsoft's FrontPage or NetObjects' Fusion products, can be similarly replaced. If these tools become DAV-capable, then they can be used against *any* DAV-enabled server.

Other ad hoc solutions, such as those based on "file upload" or doing a POST to a CGI script, can be eliminated.

HTTP/1.1 defines a solid platform for extending HTTP, which DAV uses for its benefit. In turn, this provides a solid base for other specialized protocols. A system might use DAV for moving data between a client and a server, and define new HTTP methods for invoking specialized server operations. This would avoid many pitfalls associated with trying to create a new protocol from scratch; it also provides leverage with HTTP's existing feature set.

DAV Details

As mentioned before, DAV is built upon HTTP/1.1. DAV adds new headers and methods to the protocol, along with providing additional semantics for some of the existing HTTP/1.1 methods. Part of the changes introduced by HTTP/1.1 (over 1.0) is this mechanism for extending the protocol in a clean, well-understood fashion.

Briefly, the new headers are: `DAV:`, `If:`, `Depth:`, `Overwrite:`, `Destination:`, `Lock-Token:`, `Timeout:`, and `Status-URI:`. These headers will not be discussed in detail here, but will be mentioned in association with the HTTP methods.

The new HTTP methods are: `COPY`, `MOVE`, `MKCOL`, `PROPPATCH`, `PROPFIND`, `LOCK`, and `UNLOCK`. New semantics are provided for `GET`, `PUT`, `POST`, `DELETE`, and `OPTIONS`.

COPY and MOVE

These two methods are rather obvious: they move or copy resources on the DAV server. The `Destination:` header is used to specify the target location. When collections are copied, the `Depth:` header can be used to specify how deeply the copy will be performed. The `Overwrite:` header is used to prevent overwriting, or to ensure that overwriting a target was intended.

MKCOL

This method is used to create a new collection on the server. `PUT` is only to be used for creating or updating the content of member resources.

An important point is that some servers implement `PUT` to implicitly create parent collections for a resource, if those parents do not exist. DAV has modified the semantics of `PUT` to state that all parents must exist; this ensure that parents are not accidentally created due to, say, a typographical error. A `MKCOL` must be issued to create parents before a `PUT` is performed.

PROPPATCH and PROPFIND

`PROPPATCH` is used to set, change, or delete properties on a resource. Each `PROPPATCH` operates on a single resource, but may perform an arbitrary, ordered sequence of property modifications on that resource.

`PROPFIND` is used to fetch one or more property names and/or values from one or more resources. The body of the method is used to determine the exact fetching behavior for each target resource. The `Depth:` header specifies the behavior of selecting resources when a `PROPFIND` is performed on a collection resource.

LOCK and UNLOCK

These two methods have obvious semantics: they apply and remove locks from resources. The `Depth:` header is used to select resources when a `LOCK` is performed on a resource. The `Lock-Token:` header is used by both resources: for `LOCK`, it occurs in the response and specifies the resulting lock token; for `UNLOCK`, it occurs in the request specifies the lock token to be unlocked.

The `LOCK` request must have a body, but it will not be detailed here. In short, it is used to specify the various pieces of lock state that were described earlier.

New semantics for existing methods

The new semantics that were applied to the existing methods primarily deal with their interaction with locks, with the requirement to process the `If:` header, and how to return a server's DAV capabilities in an `OPTION` request.

Some Scenarios for Use

There are many scenarios where WebDAV technology can be deployed effectively. A few samples are: collaborative authoring, a network file system, a uniform repository-access protocol, and remote/distributed software engineering projects. In each of these cases, DAV can provide a basis for communication between clients and servers.

Below are a couple detailed deployment scenarios. It is interesting to note that the scenarios are not restricted to the Internet, but may be used in various LAN, WAN, or VPN environments.

Departmental server

In this scenario, suppose there is a department with twenty staff members. Each member needs to author documents on their department's private web server.

Historically, this kind of scenario has used a central file server for collaborative authoring. More recently, and certainly with the advent of DAV, it is possibly to entirely drop the notion of a file server and work solely with a DAV-enabled web server. The web server can provide better navigation, overviews, offsite links, searching, and integration with operational systems such as finance and accounting.

By using a DAV server, the staff members can directly place and manage their documents on the web server. Each of these documents may be tagged with metadata, by using DAV properties. The staff can create auxiliary index and overview documents, or use web server features to automatically perform these functions. The server administrator can partition the URL namespace as necessary to store the documents into different backup partitions, or to spread a load over the available computer resources. In addition, the server administrator can use HTTP authentication and authorization mechanisms to restrict access to the web server on an individual basis.

Web hosting

In this scenario, there is a small Internet Service Provider ("ISP") that has 5000 users, each with personal web pages.

Typically, these users would be managing their web pages through some ad hoc system composed of FTP, NFS mounts, and possibly a FrontPage-enabled server. Every user would need to have some form of access to the system that is running the web server process.

In the DAV-enabled scenario, the server administrator can dramatically restrict access to the server. Only ISP staff would have access to the server's command shell and facilities. The ISP's users would access, create, and manage their web pages entirely through WebDAV. The web server could be configured to authenticate the user through, say, communication with the Radius server that authenticated the user's dialin process.

Setting Up mod_dav

This section is a brief overview of the processing of building, installing, and configuring the mod_dav module for the Apache web server. Detailed instructions are available at the mod_dav website[2] and within the distribution. This section will cover mod_dav 1.0.x, which is designed for the Apache 1.3.x web server. Apache 2.0 will ship with its own, newer version of mod_dav.

mod_dav is a module which adds DAV capabilities to an Apache server. Specifically, it recognizes and handles the new HTTP methods and modifies the behavior of the existing methods.

There are five basic steps:
1) Grab the tarball, unpack it, build it, and install it
2) Enable DAV for the appropriate portions of your servers' namespace
3) Prepare the repository
4) Enable locking
5) Tighten up the security

Grab, unpack, build, and install

mod_dav is similar to other Apache modules. It has a "configure" script that allows you to set a few build-time options, and it can install itself as a dynamically-loaded module or statically-linked into the Apache executable. This step is a little more difficult if you have a version of Apache prior to 1.3.9 – Expat must be installed in a "typical" location on your system or you must tell the configure process where to find it. Apache versions 1.3.9 and later include Expat, so this dependency is handled automatically for those versions.

Enabling DAV

mod_dav has a simple directive for enabling DAV within a Directory or a Location. For example:

```
Alias /gstein /home/apache/davdirs/gstein
<Location /gstein>
      DAV On
</Location>
```

In the above example, there is a location (in the URL namespace) named `/gstein`. That location, and all resources under it, becomes DAV-enabled. Further, the resources will be stored in the `/home/apache/davdirs/gstein` directory.

Preparing the repository

In the example in the previous subsection, the resources are stored into the `/home/apache/davdirs/gstein` directory. It is important to make sure that the web server process can read and **write** to that directory.

For example, if the webserver runs as user "nobody" and group "www", then the directory must allow reading and writing by that user and/or group. For example:

```
% ls -la /home/apache/davdirs/gstein
total 3
```

```
drwxr-s---    3 nobody   www    1024 Jun 25 14:32 .
drwxr-s---    3 nobody   www    1024 Jun 28 17:26 ..
-rw-r--r--    1 nobody   www     424 Jun 26 16:36 index.html
drwxr-s---    4 nobody   www    1024 Jun 26 13:05 specs
```

The repository used by mod_dav is considered *private*. Users are not allowed to modify the repository through any means other than WebDAV. This does imply that FTP cannot be used as an alternate access mechanism.

Enabling locking

mod_dav requires a lock database for recording locks on the resources. The database is specified as a filename; for example:

```
DAVLockDB /home/apache/var/DAVLockDB
```

This file will be created at runtime – it does not have to exist before using mod_dav. However, this does imply that the web server process' user/group must have write access to the directory containing the file (`/home/apache/var` in the above example).

Note that mod_dav will actually create a pair of files, such as `DAVLockDB.pag` and `DAVLockDB.dir`.

Tightening security

There are a few items needed to ensure that a DAV-enabled Apache server remains secure.

First, turn off options such as CGI for the DAV directories. Otherwise, it may be possible to place a CGI onto the server and perform an attack through that CGI. CGI, includes, and executable includes can all be turned off with the `Options` directive. For example:

```
Options None
```

Next, disable the capability for a .htaccess file to override the limitations imposed upon the DAV directory. This restriction is easily performed with:

```
AllowOverride None
```

And lastly, ensure that modifications may only be performed by authenticated users:

```
<Limit PUT POST DELETE PROPFIND PROPPATCH \\
       MKCOL COPY MOVE LOCK UNLOCK>
```

It is interesting to note that we normally want to restrict `PROPFIND` to authenticated users. The reason is that `PROPFIND` can return a listing of all the files within a DAV directory, whether they are normally discoverable through links or not. For example, there may be a PHP include file in a directory that contains a database password. It would be inadvisable to allow a user to discover that file through a `PROPFIND` and then attempt to fetch it.

In the future, the server will have finer-grained functionality and the `PROPFIND` would not need to be limited. For example, the server could hide the essential files, but allow a browser to see the "appropriate" files. This would allow a browser to use `PROPFIND` to create a nicely formatted index of the resources at a location.

Other Tools and Applications

There are a great number of tools, applications, and servers that are being created or DAV-enabled. Most of these can be found via the Projects page at webdav.org[3], but this section will cover a few of the more significant tools.

Sitecopy

Sitecopy[4] is an Open Source tool developed by Joe Orton. It allows an author to create and edit a web site on their local machine. Once the changes are complete, then sitecopy can be used to synchronize the target web server with the local copy – files will be copied up to the server, files will be moved or deleted on the server, etc. Sitecopy uses WebDAV or FTP to perform the synchronization, but the process operates much better through WebDAV. It is significantly faster and is more reliable.

Cadaver

Cadaver[5] is a very similar to the classic, command-line FTP tool, but uses the WebDAV protocol to talk to the target server. Files can be moved, copied, and deleted, on the server, by entering commands. Files can also be copied to and from the server, singly or in batch. Listings of directories can be fetched, and changing among directories is supported, both locally and remotely. Various locking operations are provided.

Cadaver is actually a very good tool for testing a server's operation.

Microsoft Office 2000

The Microsoft Office product has been DAV-enabled with the "2000" release. While FrontPage must still converse with a FrontPage-enabled server, the Word, Excel, and PowerPoint applications all support DAV operations. An author may type a URL into the "Open" dialog. The file will be locked on the server, fetched, and made available for editing. When the file is saved, it is saved directly to the web server. When the file is closed, then it is unlocked on the server.

This is a very important product in the DAV world, as Microsoft Office enjoys such a broad distribution. Many offices around the world use this product, which means they can all use DAV for interacting with their web server.

Internet Explorer 5

Through the "Web Folders" add-on package, Internet Explorer 5 can work with a DAV-enabled server. IE5 updates the Windows Explorer navigation tree to include a new node named "Web Folders". In this section, the user can directly browse a DAV server and use drag-and-drop to move files between the local and remote machines. Many of the standard Windows Explorer idioms work, such as renaming, cut/copy/paste, and deleting files.

The only drawback to this add-on package is that it does not have a "redirector" which would allow the DAV server to be mapped to a drive letter. This functionality is required for older applications to be able to open/save files on a DAV server. Without a change to the application to use the new DAV APIs in the Windows operating system, older applications can only work with files on a filesystem identified by a drive letter.

Language APIs

Various libraries, modules, or APIs exist for communicating with DAV servers. These can be found for the Python, Perl, and C/C++ programming languages. The Python API is available at http://www.lyra.org/greg/python/, the Perl API is at http://www.webdav.org/perldav/, and the C API is at http://www.webdav.org/neon/. A solid Java API has not been built and published as of September 2000.

With a robust set of APIs available to programmers, we should start seeing a larger interest in WebDAV. More experimentation and development will be possible.

Futures

Several new groups of capabilities will be added to WebDAV over the next 12 months: Advanced Collections, Searching, Access Control, and Versioning.

As these specifications stabilize, they will be incorporated into the Apache 2.0 line of mod_dav development.

Advanced Collections

The Advanced Collections functionality deals with three topics: creating references, or links, between resources; creating redirects from an (obsolete) resource location to another; and being able to specify particular orderings of the resources when returned in a `PROPFIND`. The first two items had a "last call" at the beginning of 2000, but a large amount of feedback has required a reissue of the drafts. The Working Group has been rather quiet on the subject, and an expected date for the reissue is unknown. In addition, the third item (specifying orderings) has been quiet for a rather long time.

Searching

This functionality provides for searching for resources on a DAV server that match a specific set of conditions. For example, "all resources authored by Joe Bob" or "all resources containing the word 'enterprise'." The searching specifications are reasonably stable and were updated in April 2000, but it is likely that a last call will not occur until late 2000.

Access Control

The Access Control functionality is not about *how* to do access control (e.g. NT ACLs, AFS ACLs, *nix permissions), but how to remotely *manage* the access control imposed by the DAV server. Through the WebDAV Access Control protocols, a remote user will be able to apply, change, and remove access restrictions on server resources. A working group has reformed and issued several drafts. We may see Access Control in late 2000.

Versioning

Versioning features were conceived very early in the design and requirements, but were deferred so they would not hold up the base protocol definition. The specification deals with different levels of conformance and capability, from simplistic versioning to complete configuration management. The draft is going into last call at the end of September. An RFC for versioning may be issued by the end of 2000.

The Subversion server (at http://subversion.tigris.org/) is a development project to create an Open Source versioning server using WebDAV Versioning for its network protocol.

Appendix

A "collection" is a collection, or a group, of resources. In URL namespace terms, if the URL /a/b/c is given, then /a and /a/b are collections. c may or may not be a collection in this example. Note that collections are considered to be resources, too, so that collections can contain other collections.

A "member resource" is the name given to a "leaf" resource. Basically, member resources are files or documents on the web server.

A "resource" refers to either a collection or a member resource.

This fancy naming is used for precision. In addition, "collection" is used instead of "directory" because the server might actually be using a database rather than a filesystem for storing the resources. The same logic applies to the use of "resource" and "member resource."

---

[1] The WebDAV RFC (2518) can be found at http://www.webdav.org/specs/
[2] http://www.webdav.org/mod_dav/
[3] http://www.webdav.org/projects/
[4] http://www.lyra.org/sitecopy/
[5] http://www.webdav.org/cadaver/