Internet Application Protocols

For more info see http://dsv.su.se/jpalme/abook/

Publisher
Not yet published • City

1. WebDAV

Objectives

This chapter describes an example of use of XML combined with HTTP in an application layer standard. WebDAV is a standard for managing a web site or other document collection by multiple users.

Keywords

coding
records
data structures
characters

1.1. Use of HTTP for new protocols

It has become very popular to use HTTP for other purposes than web page downloading. The reason for this is that existing software modules for HTTP can be used, and that firewalls often allow HTTP but prevents non-HTTP connections. Thus, if a non-HTTP protocol is used, one has to build holes in the firewall to allow them. This is often not need if HTTP is used.

When HTTP is used in this way, there is often a need to transfer more complex information than can easily be encoded in an HTTP header. For this purpose, XML is often used. Thus, many protocols use HTTP headers combined with XML bodies.

One example of this is SOAP, also known as “Web services”, which is a protocol for what is also known as “Remote Procedure Calls (RPC)”. A Remote Procedure Calls involves sending data to a application on another host, and getting other data back in response.

In SOAP, all the data is encoded in the bodies in XML format. In WebDAV, some information is encoded in the HTTP headers and some information in the XML bodies below the HTTP headers. HTTP headers are used for information which is easy to encode using ABNF. XML bodies are used for information which is not easy to encode using ABNF, but which is easy to encode using the more advanced encoding capabilities of XML.

A disadvantage with using HTTP as a base protocol for other application protocols is that HTTP is a very complex protocol, and has facilities which can cause a lot of difficulties, for example caching. Sometimes, application protocols based on HTTP specify a subset of HTTP to avoid such problems.

1.2. What is WebDAV

WebDAV is a standard for management of collections of documents on a remote host. It can be used for management of a web site, but also for other kinds of document collections. WebDAV is based on a model where a user

agent process is running on a personal computer, and a WebDAV server process is running on a remote server. The server holds collections. By “collections” is in WebDAV meant what is usually named “folders” or “directories” (see Figure 1).

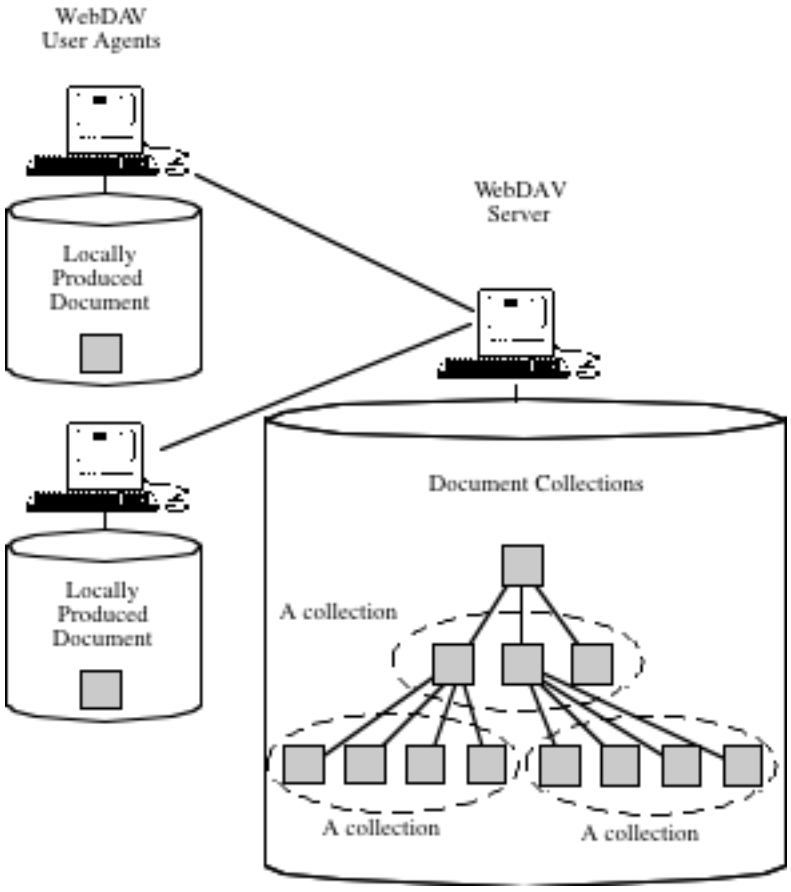

Figure 1: WebDAV model with clients, servers and document collections Files or other objects stored in a document data base are in WebDAV called *Resources*. Folders/directories of such documents are in WebDAV called *Collections* (see Figure 1).

Resources and collections can have *Properties*. Some basic properties, such as file name and last revision date, are predefined in WebDAV. Each application of WebDAV can define additional properties for its resources.

1.3. Alternatives to WebDAV

Instead of WebDAV, a common alternative solutions used by many sites is to use an HTTP-HTML-only protocol, with no User Agent process, only using an ordinary web browser as client. The advantage with WebDAV is that it allows neater clients.

Another alternative to WebDAV is FTP or SFTP, but these protocols lack certain facilities which are important when several people manage a document collection together.

1.4. Simultaneous Update Problem

One main problem which WebDAV tries to solve is the simultaneous update problem. This problem happens if two users try to update the same document on the server at the same time. There is then a risk that some of the changes to the document is lost.

Example:

User A downloads a document to modify it.
User B downloads the same document to modify it.
User A uploads the modified document.
User B uploads the modified document.

The result with this example will be that user B will overwrite the changes made by user A, so that these changes will be lost, since user B will base its changes on the document before user A has uploaded its change.

A more complex example: Suppose that that the names and the prices are stored in separate objects in the data base as shown below:

Original text:	Red Orchid:	Price $ 6.75
User A downloads this and changes it to:	Bouquet of five red orchids:	Price $ 33.75
User B downloads this and changes the descriptive text to:	Red or pink orchids	Price $ 6.75
User A saves its change:	Bouquet of five red orchids:	Price $ 33.75
User B saves its change to the description only:	Read or pink orchid:	Price $ 33.75

The result will in this example be an unintended five-fold increase in the price of the orchids.

Three common ways of handling this problem is:

Hard lock	Only one user at a time is allowed to update a certain piece of information. When one user starts to update it, a lock is set, which prevents anyone else from updating it.	Advantages: No risk of modification getting lost. Disadvantage: Another person cannot update what another person has locked, locks may inadvertently stay too long.
Soft lock	Simultaneous updaters of a certain piece of code are warned, and use procedures for avoiding problems, such as *merge* of separate changes.	Advantages: Several persons can update at the same time. Disadvantages: Merging can be done incorrectly.
Immediate update	All changes are immediately shown on screen of each simultaneous user.	Advantages: No restriction on who may update when. Disadvantages: Will work only with very fast connections between client and server.

WebDAV supports the two first methods, but does not include any merging algorithm which may be needed with a soft lock.

Hard locks are in WebDAV called *Exclusive locks*.

Soft locks are in WebDAV called *Shared locks*.

1.5. WebDAV HTTP Methods

WebDAV HTTP methods are operations which a client can perform on a server. Most of these operations can be applied either only to a resource as a whole, or to all members of a collection, usually recursively to all sublevels.

WebDAV defines the following HTTP methods:

PROPFIND: Used to get some or all properties of one or more resources. Here one can see the advantage of using XML: It is easy to return a list of resources, or even a hierarchical structure of collections and subcollections and their resources to any level.

PROPPATCH: Used to modify or remove properties of one or more resources.

MKCOL: Used to create folder/directory/collection.

GET, HEAD for collections: Standard HTTP does not specify how these operations are to operate on collections. In standard HTTP a GET on a directory (collection) can retrieve a list of files, or a start page (index.html). In WebDAV, GET and HEAD are fully defined also for collections.

POST for collections.

DELETE for resources and collections.

PUT on resources and folders/directories/collections.

COPY and MOVE on resources and folders/directories/collections.

LOCK and UNLOCK on resources and folders/directories/collections.

1.6. WebDAV Coding Methods

WebDAV encodes some of its data in the HTTP header and some in the HTTP body. The syntax of the HTTP header is specified using ABNF, the syntax of the HTTP bodies are usually specified using XML. Below is an example of part of a very simple PROPFIND operation:

HTTP header	`PROPFIND /container/ HTTP/1.1`
	`Host: www.foo.bar`
	`Content-Length: 117`
	`Content-Type: text/xml; charset="utf-8"`
		Specified using ABNF

HTTP body	`<?xml version="1.0" encoding="utf-8" ?>`
	`<D:propfind xmlns:D="DAV:">`
	`<D:prop><D:supportedlock/></D:prop>`
	`</D:propfind>`
		In XML format

Example of new or modified HTTP header fields:

Dav: Specifies that this is a WebDAV operations, and indicates version of WebDAV.

Depth: Indicates whether this operation is only to be done on a collection in itself, or recursively for all its members and submembers to unlimited depth. Usual values: "0" or "infinity".

Destination: Destination URLs for operations such as COPY and MOVE.

If: Existing HTTP IF header is extended to allow testing of WebDAV properties.

Lock-token: Unique identifier of a lock, created by the client when requesting a lock.

Overwrite: Should overwriting of existing files be allowed or not?

Time-out: Maximum duration of a lock.

Example of use of XML: A source Property for moving or copying resources:

```
<?xml version="1.0" encoding="utf-8" ?>
<D:prop xmlns:D="DAV:"
xmlns:F="http://www.foocorp.com/Project/">
   <D:source>
      <D:link>
         <F:projfiles>Source</F:projfiles>
         <D:src>http://foo.bar/program</D:src>
         <D:dst>http://foo.bar/src/main.c</D:dst>
      </D:link>
      <D:link>
         <F:projfiles>Library</F:projfiles>
         <D:src>http://foo.bar/program</D:src>
```

```
            <D:dst>http://foo.bar/src/main.lib</D:dst>
        </D:link>
        <D:link>
            <F:projfiles>Makefile</F:projfiles>
            <D:src>http://foo.bar/program</D:src>
            <D:dst>http://foo.bar/src/makefile</D:dst>
        </D:link>
    </D:source>
</D:prop>
```

Explanation of terms in the example above: “D:” = Dav namespace (defined in the WebDAV standard), “F:” = own namespace (defined by a WebDAV application), "src"=Source, "dst"=Destination.

1.7. WebDAV Extensions

The description above describes the functions of the basic WebDAV standard as defined in the standards document RFC 2518. There are additional standards documenting extensions to the basic WebDAV standard. The most important WebDAV RFCs are:

RFC 2518	Basic WebDAV: Locks, properties, collections.
RFC 3648	Versioning Extensions to WebDAV: Version history makes it possible to find and retrieve previous versions of a resource. Parallel development: The version history can fork into two different paths, for example one path for bug corrections to an old version of a resource, and another path for developing the next release with new functionality.
RFC 3648	Ordered collections - where the order is not sorting by any property values but on user-defined order. Examples: Recommended access order, Revision history order, Pages of a book, Overheads in a lecture
RFC 3744	Access Control Protocol: Access control lists, which specifies who can do what with each resource.
RFC 4316	(Experimental) Datatypes for properties (XML has rather few datatypes): Passing data type with value, schema (specifying structure of data base), Mandatory property types, Updating parts of a structured property.
RFC 4331	Quota and Size Properties.
RFC 4437	(Experimental) Redirect Reference Resources (authoring redirect commands). By Redirect is meant that someone trying to access a certain resource at a certain URL will automatically be shown another resource at another URL instead.
RFC 4709	(Informational) How WebDAV can be accessed through ordinary HTML interfaces.

By *Experimental* is meant that this is a standard for experiments to find out if this facility is to become a future standard or not. By *Informational* is meant that this is not a standard, just information for Internet application developers.

2. RSS and Podcasting

Objectives

This chapter describes the RSS and Podcasting protocols for subscribing to new information.

Keywords

RSS
podcasting
subscribing
syndication

1.8. Push and Pull

One way of understanding differences between different Internet protocols is the concepts of *Push* and *Pull*.

By *Push* is meant protocols where the sender of information has much control of what information is sent. An extremely Push-oriented protocol is e-mail. Anyone can send e-mail to anyone, if they know the recipients e-mail address. A disadvantage with this is that recipients can be overloaded with e-mail and have problems coping with it. An extreme example of this is spamming, unsolicited e-mail sent to millions of recipients, often with questionable advertising. To handle this problem, people install spam filters in their e-mail programs. But spam filers have the risk of rejecting messages which should not be rejected.

But even for e-mail which is not spamming, many people have problems handling all their e-mail. This is a natural result of a protocol where so much control is given to the senders.

Other protocols with a strong *Push* orientation are *Usenet News*, *RSS/Podcasting*, and web-based forum systems. They are, however, not as strongly Push-oriented as e-mail. With Usenet News, the recipient has some control by selecting which newsgroups to subscribe to. Usenet News also has moderated newsgroups, where one person or organization checks all messages before publishing them. This gives an added protection for recipients against unwanted information.

RSS and Podcasting are similar to moderated newsgroups, there is, usually, a person or organisation responsible for controlling what is sent out to a certain channel. A user subscribing to a channel therefore knows that only messages approved by the channel owner will be sent to it.

Example of a protocol with very strong *Pull* characteristic is ordinary web usage through search engines. The user decides what query to write to the search engine, and the search engine tries to find the best web documents corresponding to the user's query. There are also ads in search engine results, which are more *Push* oriented. But the search engines sometimes tries to finds

ads related to the user's query, giving the user some control also of which ads to see.

In Google Adwords, for example, the ad providers bids a certain price for their ads on certain search strings. These ads will only be shown to users typing the search strings which the ad providers made their bid on. When several ad providers bid on the same search string, Google will show those ads which had the highest bids, and the ads are sorted with the highest bid first in the list of ads. What was written in the previous sentence is not quite true, Google also check on how often users click on a certain ads, and ads which users often click on also can get a more prominent position in the Google search results. Ads in Google search results are clearly marked as “sponsored links”, so that the user can separate ads from regular content.

1.9. RSS and Podcasting

RSS is a protocol which allows users to subscribe to new information from certain content providers. Every time something new is available from that content provider, it is then usually automatically downloaded, but the user decides when to read the downloaded information.

The RSS protocol is very simple. The users decides which feeds they want to subscribe to. A *feed* is a web page in XML format. It contains some information about the feed itself, and then a list of available new *episodes*. For each episode, a title and/or a description is provided. A feed will usually not list all episodes, only the most recent episodes.

The user runs an RSS reader. This reader knows which episodes it has already downloaded, and downloads only the new episodes. The reader is a program running continuously in the background, and it checks for new episodes for example once every two hours.

The actual text of the episode can either be contained in the episode data itself, or (more common) the episode contains the web address of where its content can be downloaded.

The episode data can be of any format, the episode description specifies its format. RSS is commonly used for news items (shorter texts, often contained in the episode description itself) and blogs. RSS is also often used for audio

and video contents, such as broadcasts. This is known as “Podcasting”. One feed will then represent for example a regularly occurring broadcast radio program, and the RSS reader will then download these programs automatically as they are published in the feed. Some RSS readers will even automatically copy newly downloaded radio programs to an MP3 reader, if the user so wishes.

When the content is included in full in the seed, instead of in a separate document, any occurring HTML must be entity-encoded, so that for example “word” is encoded as
“>strong<word”

Here is an example, copied from Wikipedia, of a feed in RSS version 2.0 format:

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Liftoff News</title>
    <link>http://liftoff.msfc.nasa.gov/</link>
    <description>Liftoff to Space Exploration.</description>
    <language>en-us</language>
    <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>
    <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</lastBuildDate>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <generator>Weblog Editor 2.0</generator>
    <managingEditor>editor@example.com</managingEditor>
    <webMaster>webmaster@example.com</webMaster>

    <item>
      <title>Star City</title>
      <link>http://liftoff.msfc.nasa.gov/news/2003/news-starcity.asp</link>
      <description>How do Americans get ready to work with Russians
        aboard the International Space Station? They take a crash course in
        culture, language and protocol at Russia's Star City.</description>
      <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>
      <guid>http://liftoff.msfc.nasa.gov/2003/06/03.html#item573</guid>
    </item>
```

```
   <item>
    <title>Space Exploration</title>
    <link>http://liftoff.msfc.nasa.gov/</link>
    <description>Sky watchers in Europe, Asia, and parts of Alaska and
     Canada will experience a partial eclipse of the Sun on Saturday,
     May 31st.</description>
    <pubDate>Fri, 30 May 2003 11:06:42 GMT</pubDate>
    <guid>http://liftoff.msfc.nasa.gov/2003/05/30.html#item572</guid>
   </item>

   <item>
    <title>The Engine That Does More</title>
    <link>http://liftoff.msfc.nasa.gov/news/2003/news-VASIMR.asp
   </link>
    <description>Before man travels to Mars, NASA hopes to design new
     engines that will let us fly through the Solar System more quickly.
     The proposed VASIMR engine would do that.</description>
    <pubDate>Tue, 27 May 2003 08:37:32 GMT</pubDate>
    <guid>http://liftoff.msfc.nasa.gov/2003/05/27.html#item571</guid>
   </item>

   <item>
    <title>Astronauts' Dirty Laundry</title>
    <link>http://liftoff.msfc.nasa.gov/news/2003/news-laundry.asp
    </link>
    <description>Compared to earlier spacecraft, the International Space
     Station has many luxuries, but laundry facilities are not one of them.
     Instead, astronauts have other options.</description>
    <pubDate>Tue, 20 May 2003 08:56:02 GMT</pubDate>
    <guid>http://liftoff.msfc.nasa.gov/2003/05/20.html#item570</guid>
   </item>
  </channel>
</rss>
```

If you look at the RSS document above, you can see that it contains two major sections a <channel> heading with information about the channel/feed itself, and a list of <item>s describing recent episodes published on this channel.

1.10. RSS versions and standards

There is some confusion with RSS standards, because there are actually three different RSS standards, each with different versions.

Branch A: RSS 0.90, RSS 1.0 and RSS 1.1.

Branch B: RSS 0.92 and RSS 2.0. This is the most commonly used version of RSS when this is written (August 2007).

Branch C: Atom, an IETF standard published in RFC 4287.

Many RSS readers handle this by being able to receive an RSS feed in any of these three formats.

The different branches interpret the RSS acronym in different ways. In Branch A, RSS is short for *RDF Site Summary* or *Rich Site summary*. In branch B, RSS is short for *Really Simple Syndication*. One can see from the changes in interpretation that RSS is in Branch B mainly regarded as a format for *Syndication*, providing facilities to download new items as they arrive.

Below is a description of the most important XML elements in an RSS feed with the RSS 2.0 standard:

1.1.1. Information which is specified only once for a whole channel:

Element name	Description	Example
<title>	Name of the channel.	GoUpstate.com News Headlines
<link>	URL to the website for the channel.	http://www.goupstate.com/
<description>	A short description of the channel.	The latest news from GoUpsate.com, a Spartanburg Herald-Journal Web site.
<language>	Language for the channel.	en-us
<rating>	A PICS rating for the channel, specifying if it contains information unsuitable for children.	
<cloud>	Reference to a facility, where the cloud connects to your computer and tells it when there is news. If no cloud is used, the RSS reader will have to check regularly for news from the channel.	

Element name	Description	Example
<ttl>	“time to live” - tells caches how long time they can keep old cached version of the seed.	
<textInput>	Reference to a facility where people can send comments on a channel, which will then be made available to channel subscribers.	

1.1.2. Information which is specified separately for each item:

Element name	Description	Example
<title>	Title of the item.	Venice Film Festival Tries to Quit Sinking
<link>	URL from which the item can be downloaded. Note that <enclosure> is more commonly used for this.	http://nytimes.com/2004/07FEST.html
<description>	Synopsis if the item.	Some of the mot heated chatter at the Venice Film Festival this week was about the way that the arrival of the starts at the Palazzo del Cinema was being staged.
<author>	Email address of the author of the item.	
<comments>	Comments area, where comments from viewers of the item can be published.	http://www.myblog.org/cgi-local/mt/mt-comments.cgi?entry_id=290
<pubDate>	When the item was published.	

Element name	Description	Example
<guild>	A globally unique identifier of the item. Used by RSS readers to know which items they have downloaded and not downloaded>.	
<enclosure>	See below.	

1.1.3. RSS <enclosure> element

The <enclosure> element is where an RSS 2.0 item usually indicates from where it can be downloaded. Example:

```
<enclosure url="http://www.scripting.com/mp3s/
weatherReportSuite.mp3" length="12216320"
type="audio/mpeg" />
```

In addition to the *url*, the *length* and the *data type* is specified, which makes it easier for RSS readers to know how to download the item, as can be seen in the example above.