An Abstract Model for HTTP Resource State
draft-baker-http-resource-state-model-01.txt

Status of this Memo

Abstract

The documented semantics of HTTP 1.1 methods, in particular POST, are not well understood, as demonstrated by debates such as whether IPP should have used POST or a new method, how to properly bind SOAP to HTTP, and the ever more common use of POST for tunneling new protocols such as XML-RPC. This note attempts to define an abstract model for the state of HTTP URI scheme addressable resources consistent with HTTP 1.1, but hopefully more descriptive.

1. Introduction

The debate about the proper use of HTTP 1.1 [HTTP] and POST has been ongoing for quite some time. Public debate has yielded many interesting discussions and positions on the topic. Some of these include;

o "Don't Go Postal[...]", an objection to the use of POST by IPP archived at <http://www.ics.uci.edu/pub/ietf/http/draft-cohen-http-ext-postal-00.txt> and "The Use of Post", a response to same, archived at <http://www.ics.uci.edu/pub/ietf/http/draft-debry-http-usepost-00.txt>

- o Jim Whitehead's list of HTTP-extending possibilities <http://www.xent.com/FoRK-archive/feb98/0238.html>, and the minutes of a WebDAV meeting on the topic; <http://www.ics.uci.edu/pub/ietf/webdav/paloalto/minutes.html>
- o Discussion on the xml-dist-app mailing list about the correct HTTP response code to use for SOAP 1.2 faults, archived at; <http://lists.w3.org/Archives/Public/xml-dist-app/2001Jun/thread.html#15>
- o "On the use of HTTP as a Substrate for Other Protocols" an expired Internet-Draft (currently) available at; <http://www.ietf.org/internet-drafts/draft-moore-using-http-01.txt> and a response archived at; <http://lists.w3.org/Archives/Public/xml-dist-app/2000Dec/0061.html>

This note aims to define an abstract model for the state of resources identified by HTTP URIs, that is consistent with the semantics of the HTTP methods defined in [HTTP], in the hopes that it might help better explain the meaning of the HTTP methods with a holistic approach.

2. Goals and Non-goals

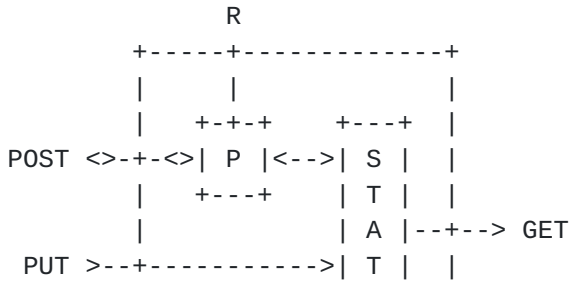
The primary goal to be met by this model is that it be consistent with the semantics of HTTP 1.1 as defined in [HTTP]. By "consistent", it is meant that it can not be used to describe invalid states or state transitions of an HTTP addressable resource.

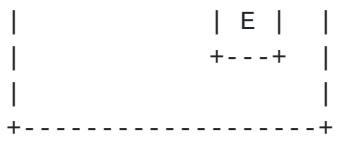
It is hoped that this model will be more or less complete with respect to HTTP 1.1 semantics, but it is explicitly a non-goal that it be so. By "complete", it is meant that the model be capable of describing all possible states or state transitions of any HTTP addressable resource.

3. The Model

In this model, as with software component models such as OpenDoc (<http://www-4.ibm.com/software/ad/opendoc/>), Java Beans v1.2 (aka BeanContext, <http://java.sun.com/products/javabeans/>), Linda (<http://www.cs.yale.edu/Linda/linda.html>) (and other tuple space systems), all resources are modeled as containers for state.

The pictorial representation of an HTTP addressable resource is as follows.





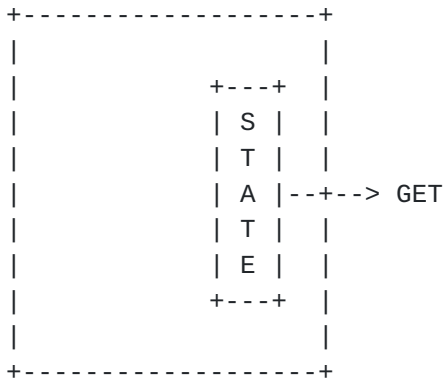
The perimeter of the resource is entirely opaque. Access to the state is provided only through the HTTP method set (shown here are the methods that operate on the resource directly). Encapsulated within the perimeter is the state of the resource, and a POST processing block "P" whose role is to take action based upon the content that is POSTed to the resource. Possible actions can include modifying the state of the resource, the creation of new subordinate resources, and manipulation of existing subordinate resources (or some combination thereof). The latter two are represented by "R" in the diagram.

3.1 Types of resources

The following is a non-exhaustive list of some of the different types of HTTP addressable resources, when classified by the way in which changes in state occur.

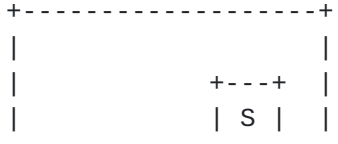
3.1.1 Read only resources

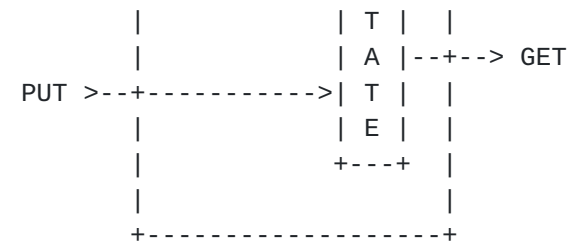
Many HTTP addressable resources today, such as the vast majority of HTML web pages, expose only the GET method. These resources are immutable with respect to HTTP clients on the Web, as no means is provided by which one may manipulate the state.



3.1.2 Simple state holding resources

These resources use the PUT method to explicitly set the state of the resource to the state represented in the body of the invocation. GET is used to subsequently retrieve that state.



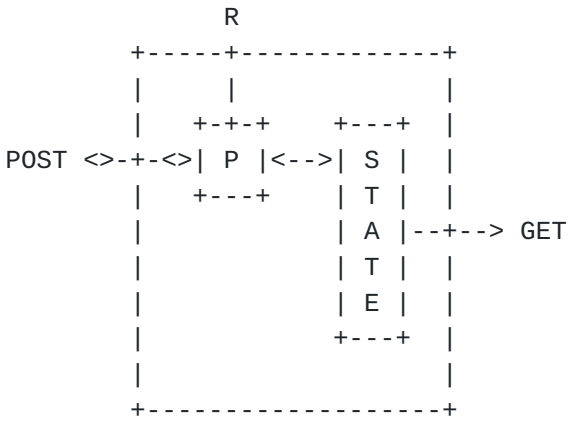


Should the resource not exist, but the web server permit it, the invocation may result in the resource being created. If this is the case, then the initial state of the resource will be that provided in the body of the PUT invocation. Any existing state that may have existed before the PUT, will be unavailable for access through further invocation of HTTP methods on this same resource.

3.1.3 Composite state resources

Resources exposing POST, when viewed through this model, and due to the definition of POST in [HTTP] section 9.5 as "accepting as a subordinate", present the notion of the state of the resource being composite; a function of the previous state of the resource, as well as the new resource representation being POSTed.

In this model, this is represented through the relationship of "P" with the state of the resource. While the job of PUT is to set the state explicitly, POST changes the state relative to the current state, where the relative change is determined by "P".



3.1.3.1 Identity-preserving composite state resources

A specialized type of "composite state resource" described in section 3.1.3, this resource has the additional property that "P" assigns a new identity to the representation of the resource being POSTed, while making those resources available through "R".

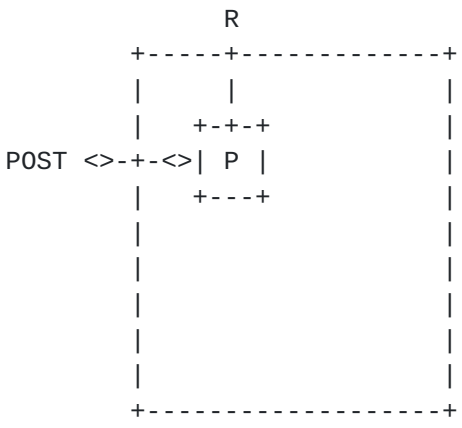
Pictorially, this is identical to 3.1.3. The sole semantic difference being that upon POSTing of the content, a 201 (Created)

response status is returned, with a Location header value being the URI of the newly created resource.

Bulletin boards or newsgroups are good examples of this type of resource, as they preserve the identity of the messages POSTed to them, thereby making them individually accessible by users of the bulletin board.

3.1.4 Processing resources

Not all resources exposing the POST method need take full advantage of the expressiveness of the composite state view. Some, like virtually all processors of POSTed HTML forms in use today, are content to maintain no state themselves (thereby making GET unnecessary), but instead simply provide their functionality through the immediate effect of processing the form and returning the results of that processing (with the possibility of creating subordinate resources via "R", though that is also not in common use).



4. Comparison with existing method definitions

The meaning of GET and PUT are well understood. The view of them implicit in this model is believed to be clearly consistent with the definition in [HTTP]. That is not the case for POST, so this section will compare the authoritative definition with the one suggested by this model.

4.1. POST

The definition of POST in [HTTP] section 9.5 describes four functions that POST is meant to provide;

"- Annotation of existing resources;

- Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;

- Providing a block of data, such as the result of submitting a form, to a data-handling process;

- Extending a database through an append operation."

The first, annotation, aims to augment the existing state of a resource with an annotated "note". For example, one might annotate an editorial with a comment. This would be represented in the composite state model as adding the POSTed comment to the resource. This may happen with or without the granting of that annotation a new identity.

The second function above, of posting a message to a forum, can be easily described with this model. The forum, be it a bulletin board, a mailing list, or similar, is a resource whose state is comprised of all articles that have been posted to it (and perhaps other information). Posting a new message to that resource augments its existing state with the new message. This is likely to be done in an identity-preserving manner, as described in section 3.1.3.1.

The third function is described by either section 3.1.4 (if there is no state maintained as a result of the processing), or by sections 3.1.3 or 3.1.3.1, should state be maintained.

The fourth function, assuming "append" means, in the case of a relational database, to add a new table, is similarly described with the database as a container for tables. A new table is added to the existing set of tables (the state of that container), not as a replacement, hence the need for POST rather than PUT.

5. Author's Address

Mark Baker
Planetfred Inc.
44 Byward Market, Suite 240
Ottawa, Ontario, CANADA. K1N 7A2
tel:+1-613-795-1818
mailto:mbaker@planetfred.com

6. Acknowledgements

The author would like to thank the following for their suggestions and corrections; Mike Dierken, Jeff Mogul, Mark Nottingham.

7. References

[HTTP] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

Appendix A. Changes

-01; added ascii art. Found that I needed to simplify the model to make a reasonably simple diagram, so that worked out well despite requiring

a rewrite of much of the content. Added a section about consistency vs. completeness.