

ColdFusion Foundations: HTTP



Mosh Teitelbaum

mosh.teitelbaum@evoch.com

evoch, LLC

HTTP: What Is It?

- Officially

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. - *RFC 2616*

- Unofficially

A protocol that defines how web applications communicate

- Huh?

It's how the web works

HTTP: Selling Points

- **Universally Accepted**

If your application sticks to the specification, it will work with every other application that sticks to the specification.

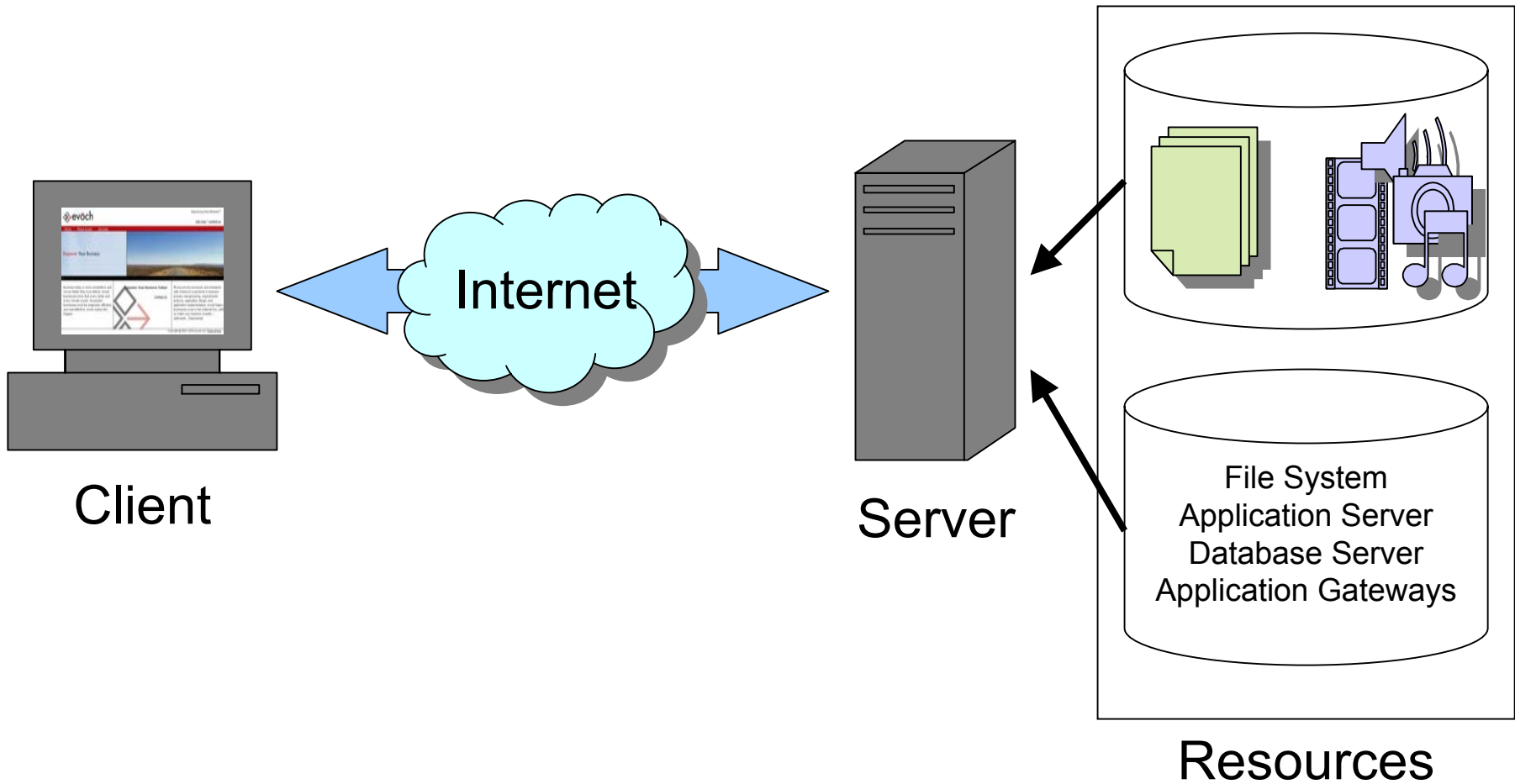
- **Reliable**

It's layered on top of TCP for reliable transmission and in-order receipt of data.

- **Simple**

Simple transactions and message format are easy to understand, program, and debug.

HTTP: Who's Talking



HTTP: Transactions

- All HTTP client and server transactions follow these steps:
 1. Client connects to server
 2. Client sends request message identifying a resource
 3. Server sends response message
 4. Server disconnects from client
- HTTP 1.1 assumes multiple requests and responses per transaction

HTTP: Uniform Resource Identifiers (URI)

- Identify web resources
- Come in 2 forms:
 - URN – Uniform Resource Name
 - URL – Uniform Resource Locator
- URNs uniquely identify a resource
- URLs define how to locate a resource
- Most URIs are of the URL variety

HTTP: URLs

<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>

scheme	The protocol used to access a resource
user	The username required to access a resource
password	The password required to access a resource
host	The name or IP Address of the server hosting the resource
port	The port number on which the host server is listening
path	The local name/path for the resource
params	Specifies input parameters for some schemes
query	Specifies input parameters for some schemes
frag	A name for a portion of a resource. Not sent to the server.

HTTP: Messages

- HTTP Messages consist of:

1. Start-line

Describes the intent of the message

2. Message-Header(s)

One or more headers describing the message or the Entity-body followed by a blank line

3. Entity-body

The actual data of the message

HTTP: Request Messages

<method> <URL> <version>

<headers>

<body>

GET /Doc/index.cfm HTTP/1.1
Accept: text/* Connection: close

POST /action.cfm HTTP/1.0
Content-type: application/x-www-form-urlencoded Content-length: 23
Name=Mosh&Company=evoch

HTTP: Response Messages

<version> <status> <reason>

<headers>

<body>

HTTP/1.0 200 OK
Content-type: text/html Content-length: 19
Hello World!

HTTP/1.0 302 Found
Set-cookie: Name="Mosh" Location: /success.cfm

HTTP: Message Components

Component	Description
method	The action to perform on the resource
URL	The URL of the resource
version	The HTTP version being used
status	A 3 digit status code describing the result of the request
reason	A string describing the result of the request. The string is not standardized and should be ignored by applications.
headers	Zero or more headers providing meta-data about the message. The header block is terminated by a blank line containing just a CRLF.
body	An optional block of data

HTTP: Methods

The method tells the server what action the client would like it to take

Method	Description
OPTIONS	Find out what methods work on the specified resource
GET	Retrieve the specified resource
HEAD	Retrieve only the headers of the specified resource
POST	Send data to the specified resource
PUT	Create or overwrite a resource on the server
DELETE	Delete a resource from the server
TRACE	Find out what the server sees when it receives a request
CONNECT	Reserved for use with proxy servers

HTTP: Status Code Ranges

The status code informs the client what happened as a result of its request

Range	Defined	Category	Meaning
1xx	100 – 101	Informational	An interim response always followed by another action
2xx	200 – 206	Successful	The request was successful
3xx	300 – 305	Redirection	Further action is required on the part of the client
4xx	400 – 415	Client Error	The client erred in some way
5xx	500 – 505	Server Error	The server erred or cannot complete the request

HTTP: Common Status Codes

Some of the more common status code are:

Status Code	Seen when...
200	the request was successful
302	the browser is instructed to redirect to another URL
403	you try to access a protected area without valid credentials
404	the resource you requested does not exist
500	the server encounters an internal error

HTTP: Header Classifications

Headers add information to the request or response.

Classification	Definition
----------------	------------

General	Can be used in both requests and responses and pertain to the message itself, not to the entity being sent/received.
Request	Add additional information about the request and/or the client
Response	Add additional information about the response and/or the server
Entity	Defines information about the entity and/or the resource
Custom	Headers not defined by HTTP and are usually specific to an application

HTTP: Common Headers

Some of the more common headers are:

Header	Classification	Description
Accept	Request	Specifies acceptable response media types
Authorization	Request	Specifies authentication credentials
Cache-Control	General	Specifies directives to caching mechanisms
Content-Length	Entity	Indicates the size of the body
Content-Type	Entity	Indicates the media type of the body
Date	General	Specifies the date/time the message was sent
Last-Modified	Entity	Specifies the date/time the entity was last modified
Referer [sic]	Request	Specifies the referrer URI

HTTP: Sample Transaction #1 - GET

GET /http/hello.html HTTP/1.0

HTTP/1.0 200 OK
Content-type: text/html
Content-length: 90
<HTML>
<HEAD>
<TITLE>Hello World</TITLE>
</HEAD>
<BODY>
Hello World
</BODY>
</HTML>

HTTP: Sample Transaction #2 - HEAD

HEAD /http/hello.html HTTP/1.0

HTTP/1.0 200 OK
Content-type: text/html
Content-length: 90

HTTP: Sample Transaction #3 - POST

POST /http/add.cfm HTTP/1.0
Content-type: application/x-www-form-urlencoded Content-length: 13
num1=3&num2=5

HTTP/1.0 200 OK
Content-type: text/html Content-length: 73
<HTML> <HEAD> <TITLE>Add</TITLE> </HEAD> <BODY> 3 + 5 = 8 </BODY> </HTML>

ColdFusion: Exposing HTTP

- CGI Variables

CGI.HTTP_* variables allow us to see headers sent in the request message. Other CGI scope variables let us see other information about the request message.

- HTTP Server Tags and Functions

These tags and functions enhance our ability to control how the HTTP server responds to the request.

- HTTP Client Tags

These tags allow ColdFusion to act as an HTTP Client and to send requests to HTTP Servers and parse the responses.

ColdFusion: CGI Variables

- **CGI Server Variables**
CGI.SERVER_* (and other) variables expose information about the web server software and the server environment.
- **CGI Client Variables**
CGI.HTTP_* variables expose header names and values as passed by the browser.
- **CGI Client Certificate Variables**
CGI.CERT_* variables expose information about client SSL certificates.

ColdFusion: CGI Client Variable Example

```
GET /http/cgi.cfm HTTP/1.0
```

```
User-agent: Mosh App 2000
```

```
Bogus: Fake
```

```
Authorization: Who Knows?
```

```
HTTP/1.0 200 OK
```

```
Content-type: text/html
```

```
Page-Completion-Status: Normal
```

```
AUTH_TYPE = Who<BR>
```

```
HTTP_AUTHORIZATION = Who Knows?<BR>
```

```
HTTP_BOGUS = Fake<BR>
```

```
HTTP_USER_AGENT = Mosh App 2000<BR>
```

```
PATH_INFO = /http/cgi.cfm<BR>
```

```
REMOTE_HOST = 127.0.0.1<BR>
```

```
REQUEST_METHOD = GET<BR>
```

```
SERVER_PORT = 80<BR>
```

```
...
```

ColdFusion: HTTP Server Tags

- **CFHEADER**

Generates custom HTTP response headers to return to the client.

- **CFCOOKIE**

Defines cookie variables, including expiration and security options.

- **CFLOCATION**

Opens a ColdFusion page or HTML file. (But not really)

- **CFCONTENT**

Defines the MIME type returned by the current page. Optionally, lets you specify the name of a file to be returned with the page.

ColdFusion: CFHEADER Tag

Creates a new header, in the header section of the response message, with the specified name and value.

Attribute	Description
Name	Required if you do not specify the statusCode attribute. A name for the header.
Value	Optional. A value for the HTTP header. This attribute is used in conjunction with the name attribute.
StatusCode	Required if you do not specify the name attribute. A number that sets the HTTP status code.
StatusText	Optional. Text that explains the status code. This attribute is used in conjunction with the statusCode attribute.

ColdFusion: CFHEADER Example 1

```
<CFHEADER NAME="name" VALUE="value">
```

Creates a new header, in the header section of the response message, with the specified name and value.

GET /http/cfheader1.cfm HTTP/1.0

HTTP/1.0 200 OK
Content-type: text/html
Page-Completion-Status: Normal
Bogus: Fake
Some Text

ColdFusion: CFHEADER Example 2

```
<CFHEADER STATUSCODE="code" STATUSTEXT="text">
```

Sets the Status and Reason in the Start-line of the response message.

GET /http/cfheader2.cfm HTTP/1.0

HTTP/1.0 299 Kinda So-So
Content-type: text/html
Page-Completion-Status: Normal
Bogus: Fake
Dig the custom status code and text

ColdFusion: CFCOOKIE Tag

Defines cookie variables, including expiration and security options.

Attribute	Description
Name	Required. The name of the cookie variable.
Value	Optional. The value assigned to the cookie variable.
Expires	Optional. Schedules the expiration of a cookie variable. Can be specified as a date, number of days, "Now", or "Never".
Secure	Optional. Yes or No. Specifies that the variable must transmit securely.
Path	Optional. Specifies the URL within a domain to which this cookie applies.
Domain	Optional. Specifies the domain for which the cookie is valid and to which the cookie content can be sent.

ColdFusion: CFCOOKIE Example

```
<CFCOOKIE NAME="name" VALUE="value">
```

Defines cookie variables, including expiration and security options.

GET /http/cfcookie.cfm HTTP/1.0

HTTP/1.0 200 OK
Content-type: text/html
Page-Completion-Status: Normal
Set-Cookie: NAME=Val; path=/;
Notice the extra "Set-Cookie" cookie header.

ColdFusion: CFLOCATION Tag

Opens a ColdFusion page or HTML file. But it doesn't really do that. It actually redirects the client to another HTTP resource.

Attribute	Description
URL	The URL of the HTML file or CFML page to open.
AddToken	Optional. Yes or No. clientManagement must be enabled [in the CFAPPLICATION tag]. Yes appends client variable information to the URL you specify in the url attribute.

ColdFusion: CFLOCATION Example

```
<CFLOCATION URL="url" ADDTOKEN="No">
```

Opens a ColdFusion page or HTML file. But it doesn't really do that. It actually redirects the client to another HTTP resource.

GET /http/cflocation.cfm HTTP/1.0

HTTP/1.0 302 Object Moved
Content-type: text/html
Content-length: 135
Location: resource.cfm
<head>
<title>Document Moved</title>
</head>
<body><h1>Object Moved</h1>
This document may be found
here</body>

ColdFusion: CFCONTENT Tag

Defines the MIME type returned by the current page. Optionally, lets you specify the name of a file to be returned with the page.

Attribute	Description
Type	Required. Defines the File/ MIME content type returned by the current page.
DeleteFile	Optional. Yes or No. Yes deletes the file after the download operation. Defaults to No. This attribute applies only if you specify a file with the file attribute.
File	Optional. The name of the file being retrieved.
Reset	Optional. Yes or No. Yes discards output that precedes the call to cfcontent. No preserves the output that precedes the call. Defaults to Yes. The reset and file attributes are mutually exclusive. If you specify a file, the reset attribute has no effect. See Note.

ColdFusion: CFCONTENT Example 1

`<CFCONTENT TYPE="type">`

Defines the MIME type returned by the current page. Optionally, lets you specify the name of a file to be returned with the page.

GET /http/cfcontent1.cfm HTTP/1.0

HTTP/1.0 200 OK
Content-type: text/plain
Content-length: 56
This HTML code should be displayed as plain text.

ColdFusion: HTTP Server Functions

- **getHttpRequestData()**
Makes HTTP request headers and body available to CFML pages. Does not take arguments. Returns a ColdFusion structure. GetHttpRequestData is especially useful for capturing SOAP request data, which can be delivered in an HTTP header.
- **getHttpTimeString()**
This function takes one argument, a ColdFusion date/time object, and returns the time formatted as a string according to the HTTP standard described in RFC1123.
- **urlDecode()**
Decodes a URL-encoded string.
- **urlEncodedFormat()**
Returns a URL-encoded string.

ColdFusion: HTTP Client Tags

- **CFHTTP**

Performs GET and POST to upload files or post a form, cookie, query, or CGI variable directly to a specified server (CFMX 6 and lower). CFMX 6.1 added the ability to perform other HTTP methods and to better control the request message.

- **CFHTTPPARAM**

Allowed inside CFHTTP tag bodies only. Required for CFHTTP POST operations. Optional for all others. Specifies parameters to build an HTTP request.

- **CFINVOKE**

Invokes a Web Service. Introduced in CFMX.

- **CFINVOKEARGUMENT**

Passes the name and value of a parameter to a web service. This tag is used within the CFINVOKE tag. Introduced in CFMX.

ColdFusion: CFHTTP Tag

Generates an HTTP request and handles the response from the server.

Attribute	Description
Url, Port	Address/port of the resource which will handle the request.
Method	The HTTP method to perform.
UserAgent	Text to put in the user agent request header.
Redirect	If the response header includes a Location field, determines whether to redirect execution to the URL specified in the field.
ProxyServer	Used to send the request to a proxy server. See also ProxyPort, ProxyUser, and ProxyPassword
Username	Used to authenticate against a server. See also Password.
File, Path	Saves response contents to a file. See all ResolveUrl.
Name	Turns the response contents into a CF query object. See also Columns, FirstRowAsHeaders, Delimiter, and TextQualifier.

ColdFusion: CFHTTPPARAM Tag

Specifies parameters to build an HTTP request.

Attribute	Description
Type	Information type: Header, CGI, Body, XML, File, URL, FormField, or Cookie.
Name	Variable name for data that is passed.
Value	Value of the data that is sent.
File	The absolute path to the file that is sent in the request body.
Encoded	Specifies whether to URLEncode the form field or header.
MimeType	Specifies the MIME media type of the file contents.

ColdFusion: No Content Example

The 204 Status Code signifies that the request completed successfully but that there is no content being sent back in the response message. Sending back a 204 causes the web browser to stay on the current page, without updating it's contents.

```
<CFHEADER STATUSCODE="204" STATUSTEXT="No Content">
```

ColdFusion: Simulating CFContent Example

CFCONTENT is a powerful tag that, because of its ability to access and delete files from a server, is often disabled in shared hosting environments. However, using the CFHEADER (and, optionally, the CFINCLUDE) tag, you can simulate the functionality of CFCONTENT.

```
<CFHEADER NAME="Content-Type" VALUE="text/plain">
```

```
<CFINCLUDE TEMPLATE="someFile.txt">
```

Unfortunately, this will only work for text (non-binary) data files.

HTTP: Advanced Capabilities

- **Security**
HTTP provides for several forms of security: Digital Certificates, Digital Signatures, Authentication, and HTTPS.
- **Internationalization**
HTTP allows for the specification of page-specific languages and character sets
- **Web Application Support**
HTTP plays nicely with various kinds of other web applications including web bots such as spiders), proxy servers, caching servers, gateways, and tunnels.
- **Transmission Optimizations**
HTTP allows for web clients to optimize resource downloads by requesting multiple resources in a single HTTP transaction and by If-* headers to retrieve a resource only if certain conditions are met.

HTTP: Resources

- RFCs from <http://www.ietf.org/rfc/rfc####.txt>:
 - rfc1945.txt – “Hypertext Transfer Protocol -- HTTP/1.0”
 - rfc2616.txt – “Hypertext Transfer Protocol -- HTTP/1.1”
 - rfc1867.txt – “Form-based File Upload in HTML”
 - rfc1738.txt - “Uniform Resource Locators (URL)”
 - rfc2396.txt - “Uniform Resource Identifiers (URI): Generic Syntax”
 - rfc2617.txt - “HTTP Authentication: Basic and Digest Access Authentication”
 - rfc1521.txt – “MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies”
 - rfc2045.txt - “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”
 - rfc2965.txt - “HTTP State Management Mechanism” – Cookie standard
- <http://www.w3.org/Protocols/HTTP/AsImplemented.html> - The HTTP 0.9 protocol
- http://home.netscape.com/newsref/std/cookie_spec.html - NS Cookie standard

Closing

- Questions?
- Contact Info
 - Mosh Teitelbaum
 - evoch, LLC
 - mosh.teitelbaum@evoch.com
 - <http://www.evoch.com/>
- Extras
 - Reserved Characters in URLs
 - Defined Status Codes
 - Defined Headers

Extras: Reserved Characters in URLs

The following characters are reserved and/or restricted in URLs:

Character	Status
%	Escapes encoded characters
/	Reserved for splitting or defining path segments
# ? ; : @ & =	Reserved for separating URL components
\$, +	Reserved
{ } \ ^ ~ [] ‘	Restricted because some gateways don't play nice with them
< > “	Not safe because they are often used to delimit URLs
ASCII 0 – 31, 127	Restricted because they are non-printable characters
ASCII 127+	Restricted because they fall outside of the US-ASCII 7-bit character set

Extras: Defined Status Codes

Code	Meaning	Code	Meaning	Code	Meaning
100	Continue	305	Use Proxy	411	Length Required
101	Switching Protocols	306	(Unused)	412	Precondition Failed
200	OK	307	Temporary Redirect	413	Request Entity Too Large
201	Created	400	Bad Request	414	Request-URI Too Long
202	Accepted	401	Unauthorized	415	Unsupported Media Type
203	Non-Authoritative Information	402	Payment Required	416	Requested Range Not Satisfiable
204	No Content	403	Forbidden	417	Expectation Failed
205	Reset Content	404	Not Found	500	Internal Server Error
206	Partial Content	405	Method Not Allowed	501	Not Implemented
300	Multiple Choices	406	Not Acceptable	502	Bad Gateway
301	Moved Permanently	407	Proxy Authentication Required	503	Service Unavailable
302	Found	408	Request Timeout	504	Gateway Timeout
303	See Other	409	Conflict	505	HTTP Version Not Supported
304	Not Modified	410	Gone		

Extras: Defined Headers

Accept

Accept-Charset

Accept-Encoding

Accept-Language

Accept-Ranges

Age

Allow

Authorization

Cache-Control

Connection

Content-Encoding

Content-Language

Content-Length

Content-Location

Content-MD5

Content-Range

Content-Type

Date

Etag

Expect

Expires

From

Host

If-Match

If-Modified-Since

If-None-Match

If-Range

If-Unmodified-Since

Last-Modified

Location

Max-Forwards

Pragma

Proxy-Authenticate

Proxy-Authorization

Range

Referer

Retry-After

Server

TE

Trailer

Transfer-Encoding

Upgrade

User-Agent

Vary

Via

Warning

WWW-Authenticate