Overall Functioning of HTTP Clients: Complex HTTP Interactions

Web Engineering

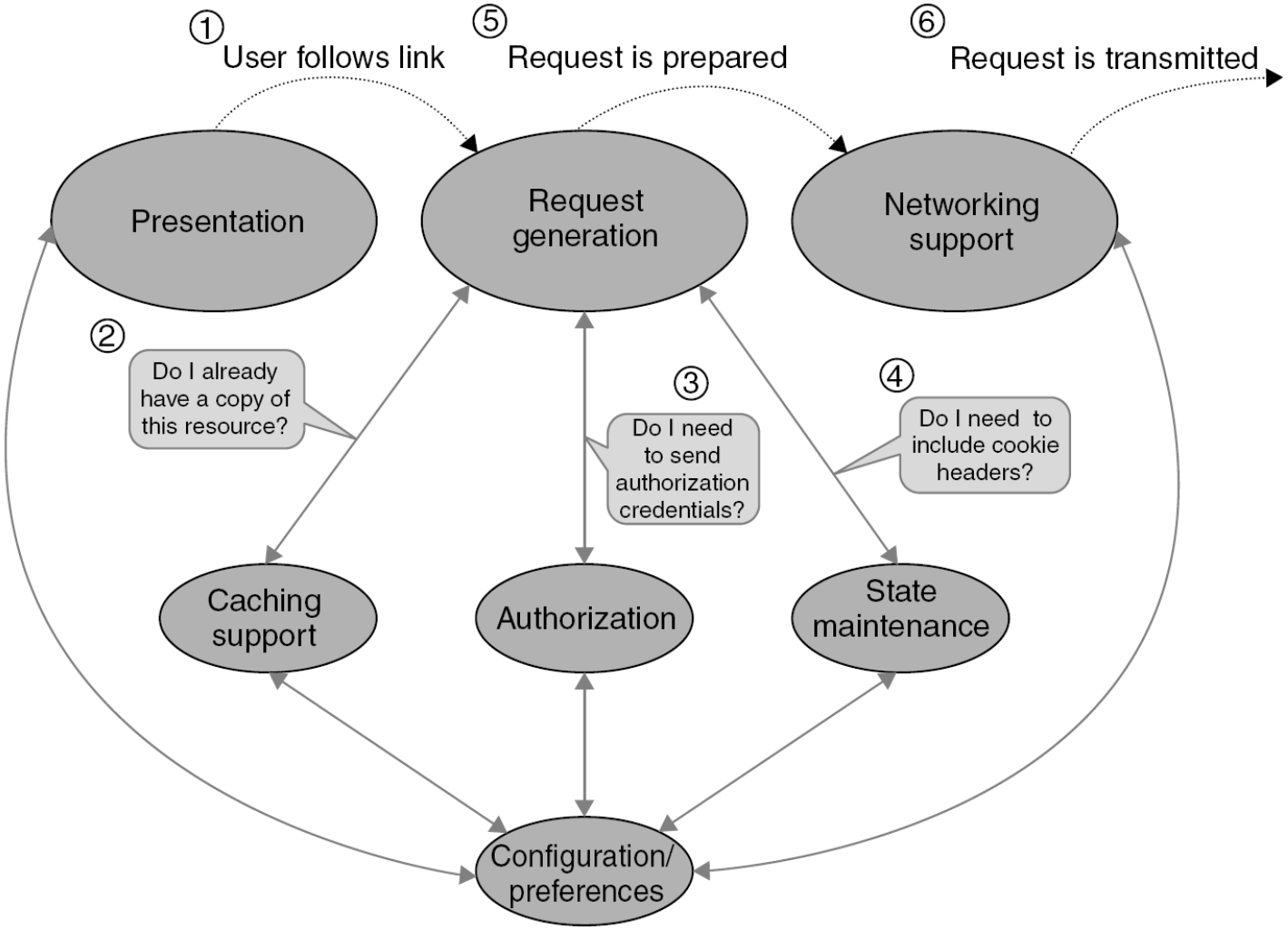
utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Complex HTTP Interactions

Cache

Authorization

Cookies

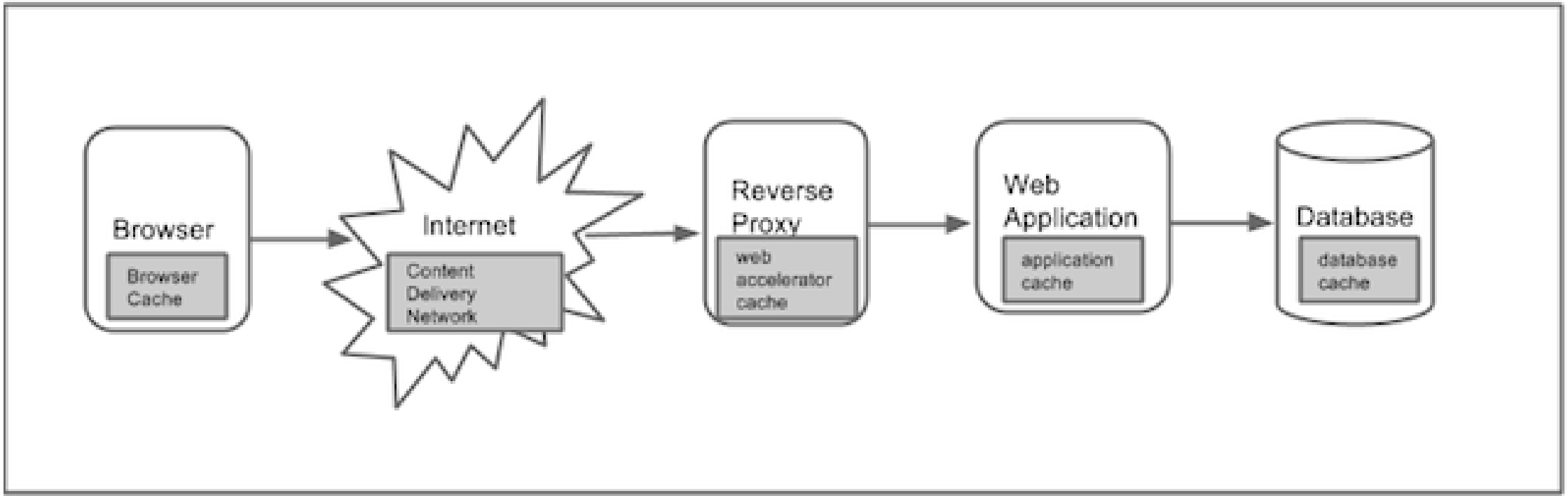


Web Cache… what is it?

Web caching is temporary storage on the hard disk of web pages, images and other documents and files using caching techniques to reduce available bandwidth usage, increase access speed, among other advantages.

From Wikipedia, the free encyclopedia

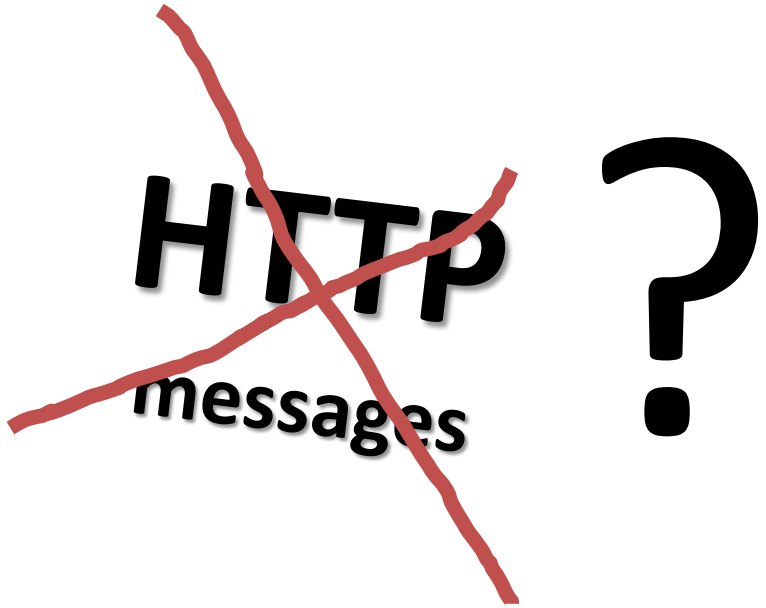
Cache Types



Server-side caching

Client-side caching

Server-side Caching



Client-side Caching

Mechanisms involved:

Request Generation module

Response Generation module

…for saving retrieved resourses

Client-side Caching – HTTP Headers

If-Modified-Since

Cache-Control

no-cache

Date

Expires

No-store

private

public

304 Not Modified

max-age

Last-Modified

Cacheable Response Codes

Code	Description	Explanation
200	Ok	success
203	Non-authoritative information	Same as 200, but sender has reason to believe that the entity headers are different from those the origin server would send
206	Partial content	Similar to 200, but response to a "range" request. Cacheable if the cache supports range requests.
300	Multiple choices	Response includes choices from which user could make a selection
301	Moved permanently	New URL is in the response headers
410	Gone	Requested resource moved permanently from origin server

Cacheable Request Methods

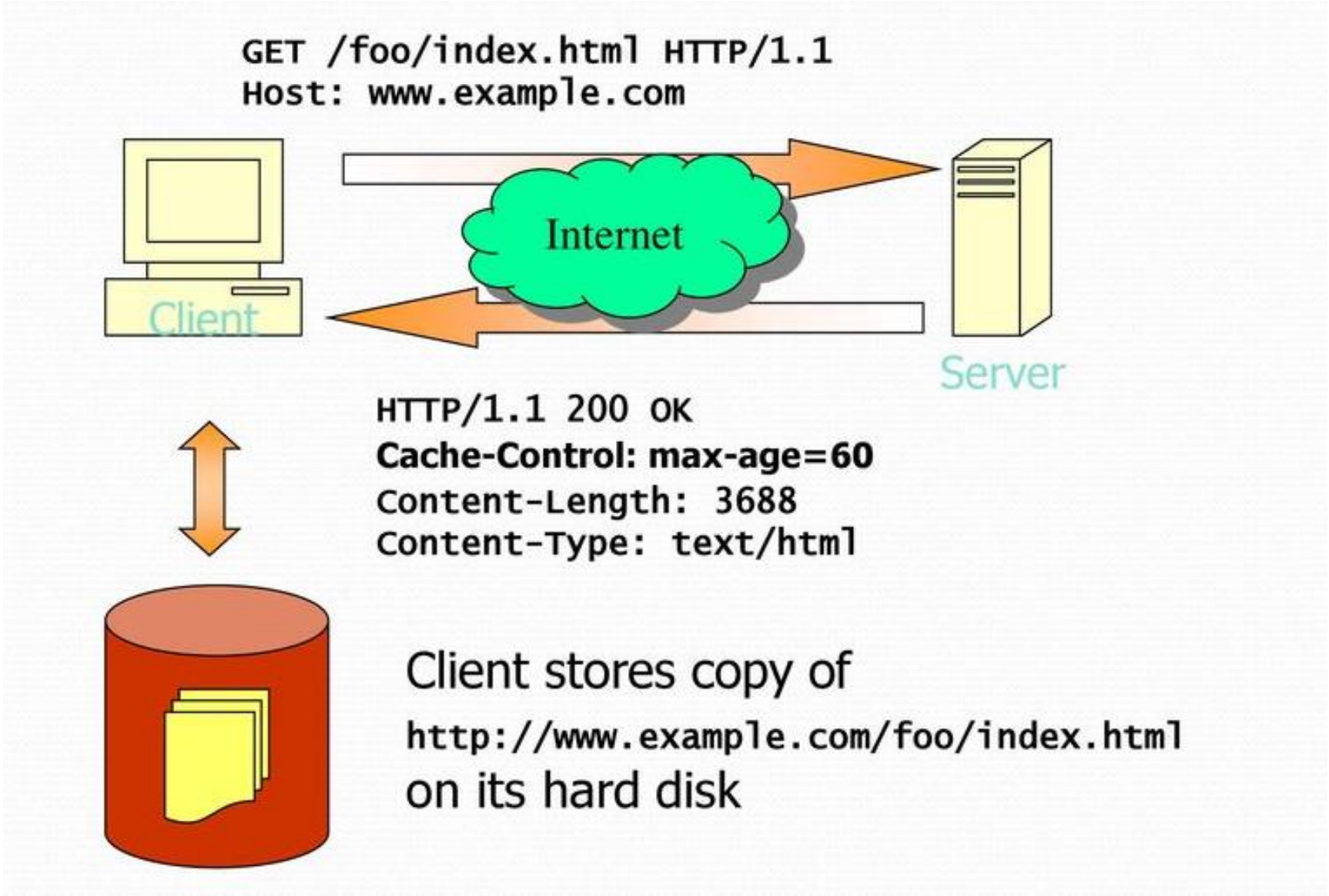
Request method	Cacheable ?
GET	Yes, by default
POST	Uncachable by default, cacheable if Cache-control headers allow
HEAD	May be used to cache prev updated entry
PUT	No
DELETE	No
OPTIONS	No
TRACE	No

Client-side Caching - examples

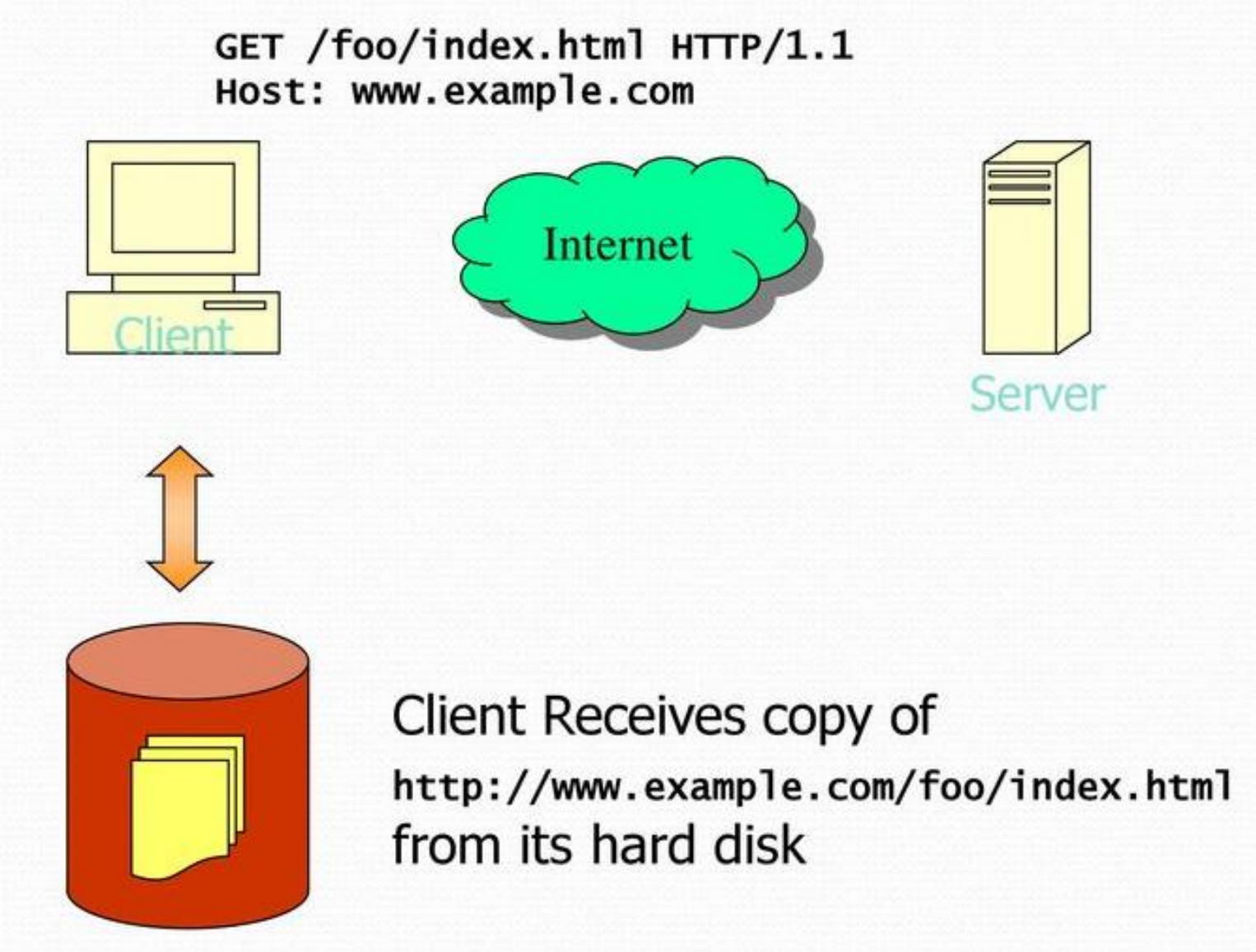
```
HTTP/1.1 200 OK
Date: Sun, 13 May 2001 12:36:04 GMT
Content-Type: image/jpeg
Content-Length: 34567
 ...
Cache-Control: no-cache
Pragma: no-cache

 ...
```

```
HTTP/1.1 200 OK
Date: Sun, 13 May 2001 12:36:04 GMT
Content-Type: image/jpeg
Content-Length: 34567
Cache-Control: private
Expires: Mon, 14 May 2001 12:36:04 GMT
Last-Modified: Sun, 13 May 2001 12:36:04 GMT

 ...
```

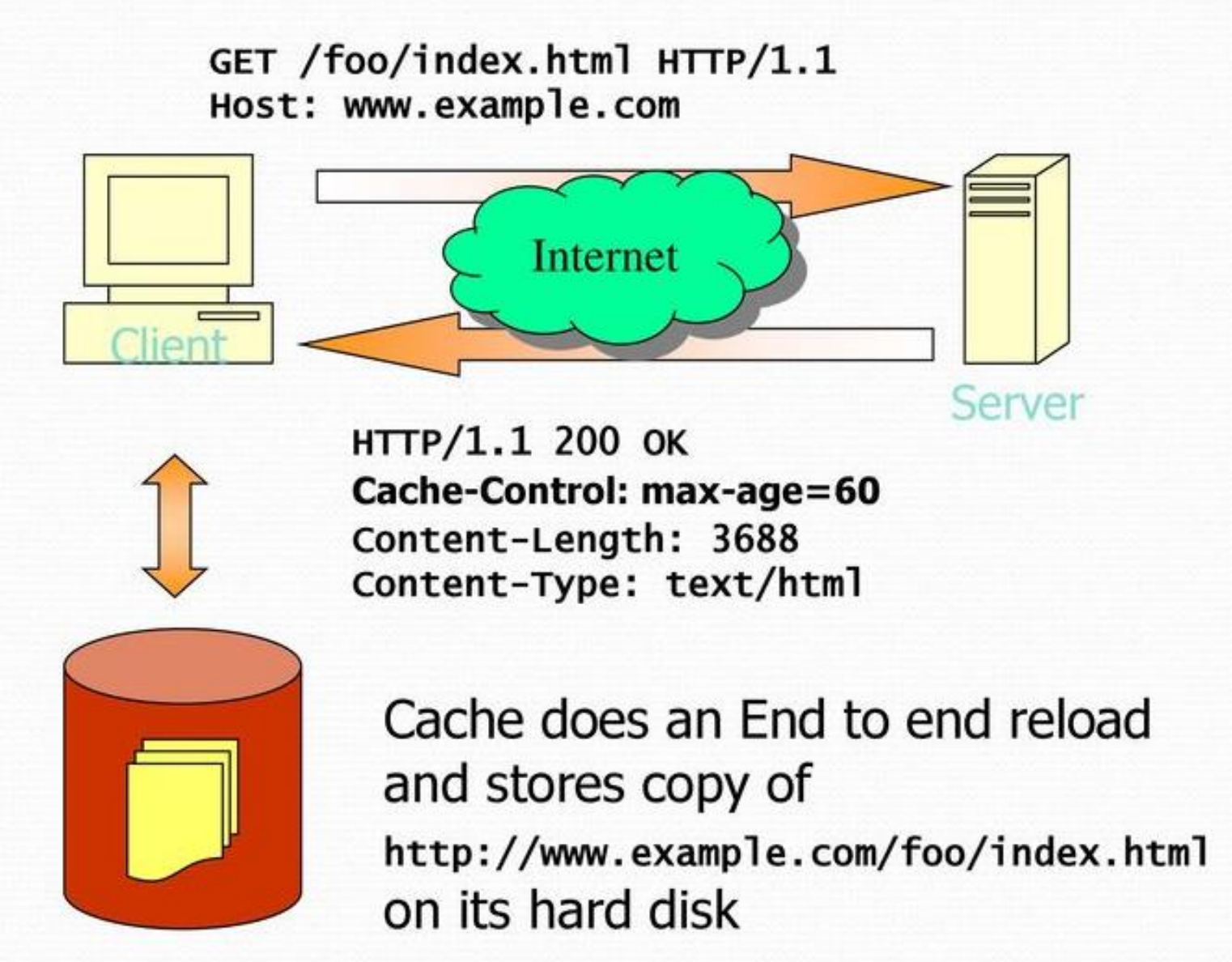
Example 1 - Client caches a response



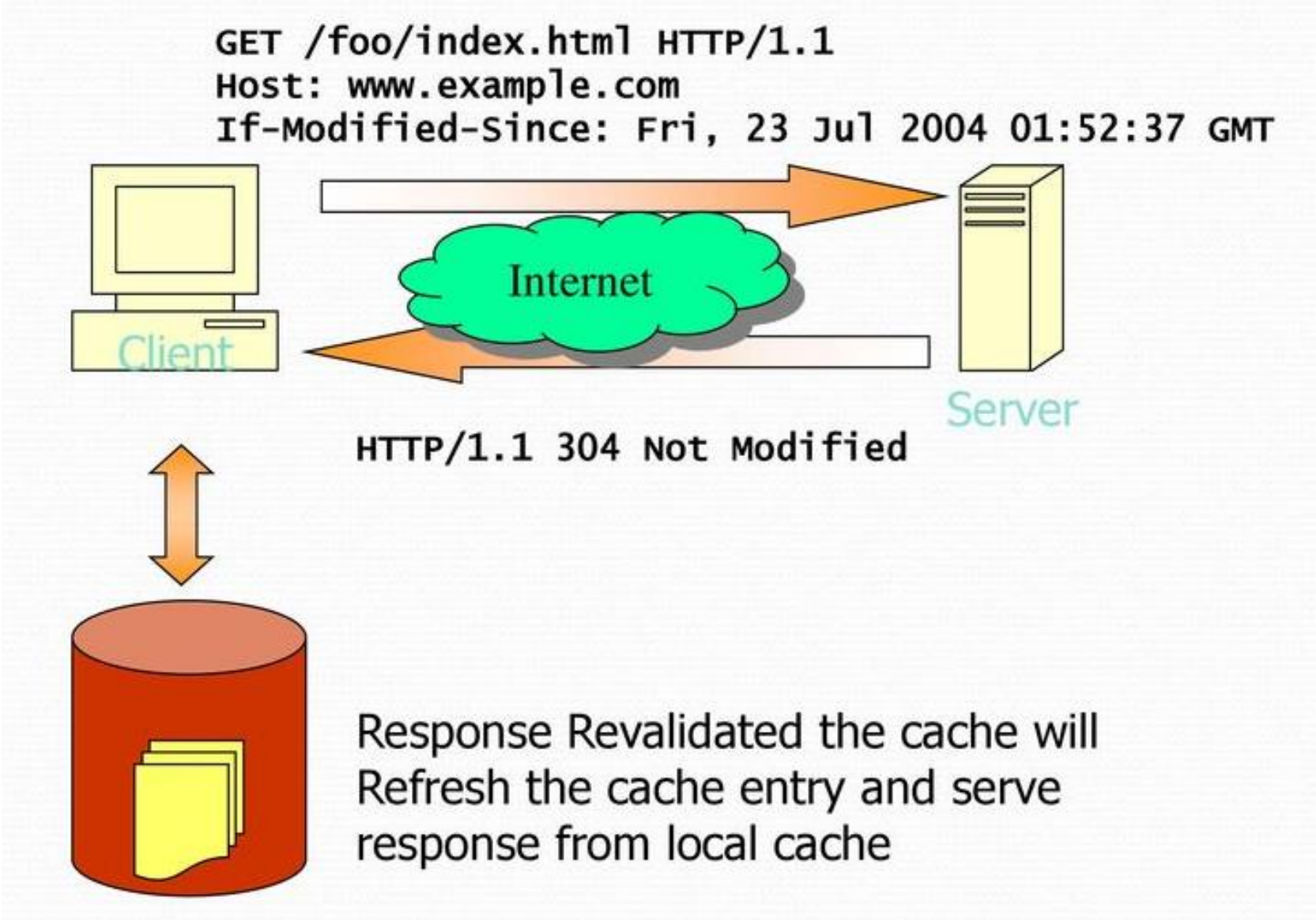
Example 2 - Client cache hit



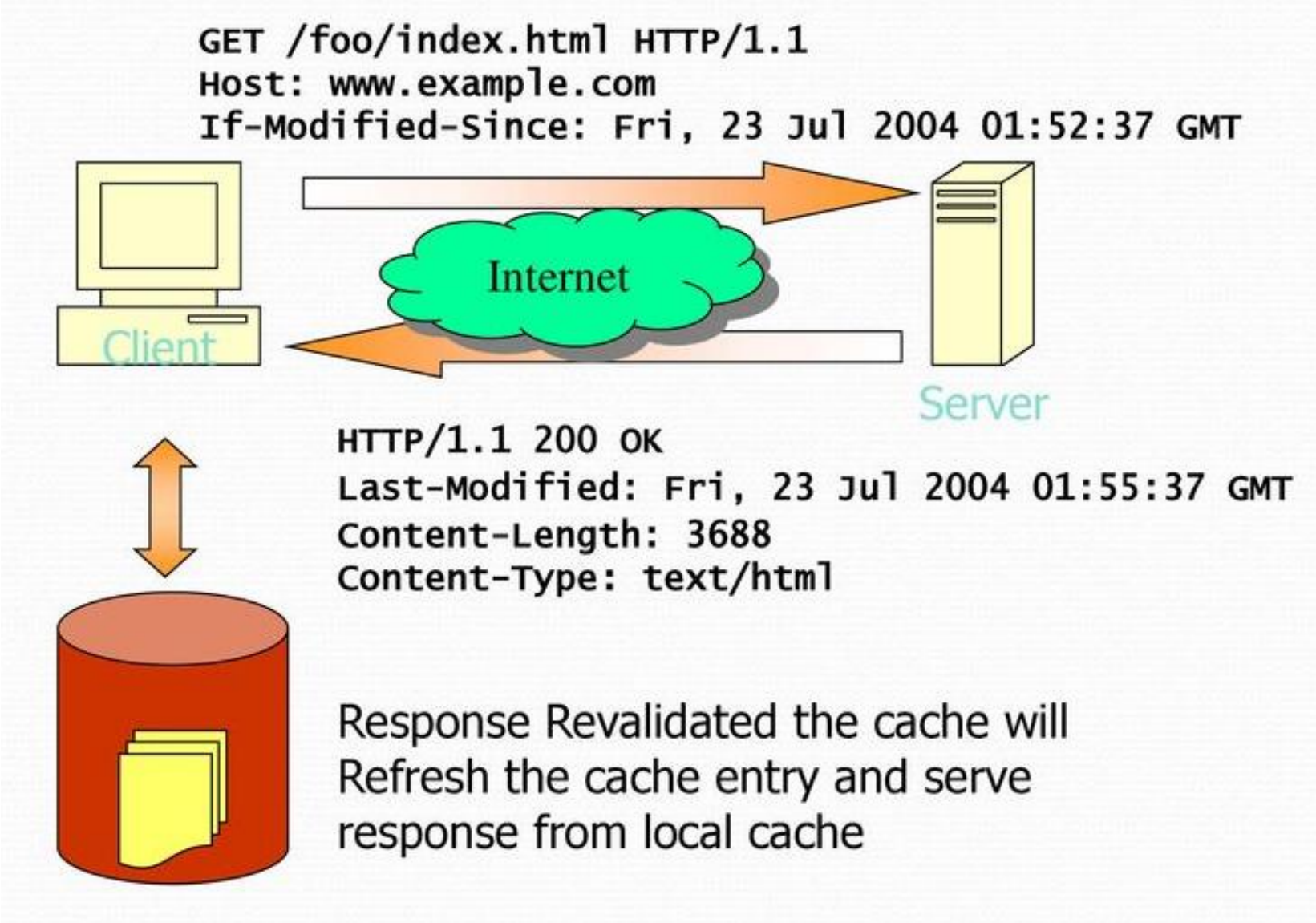
Example 3 – Cached entry expires



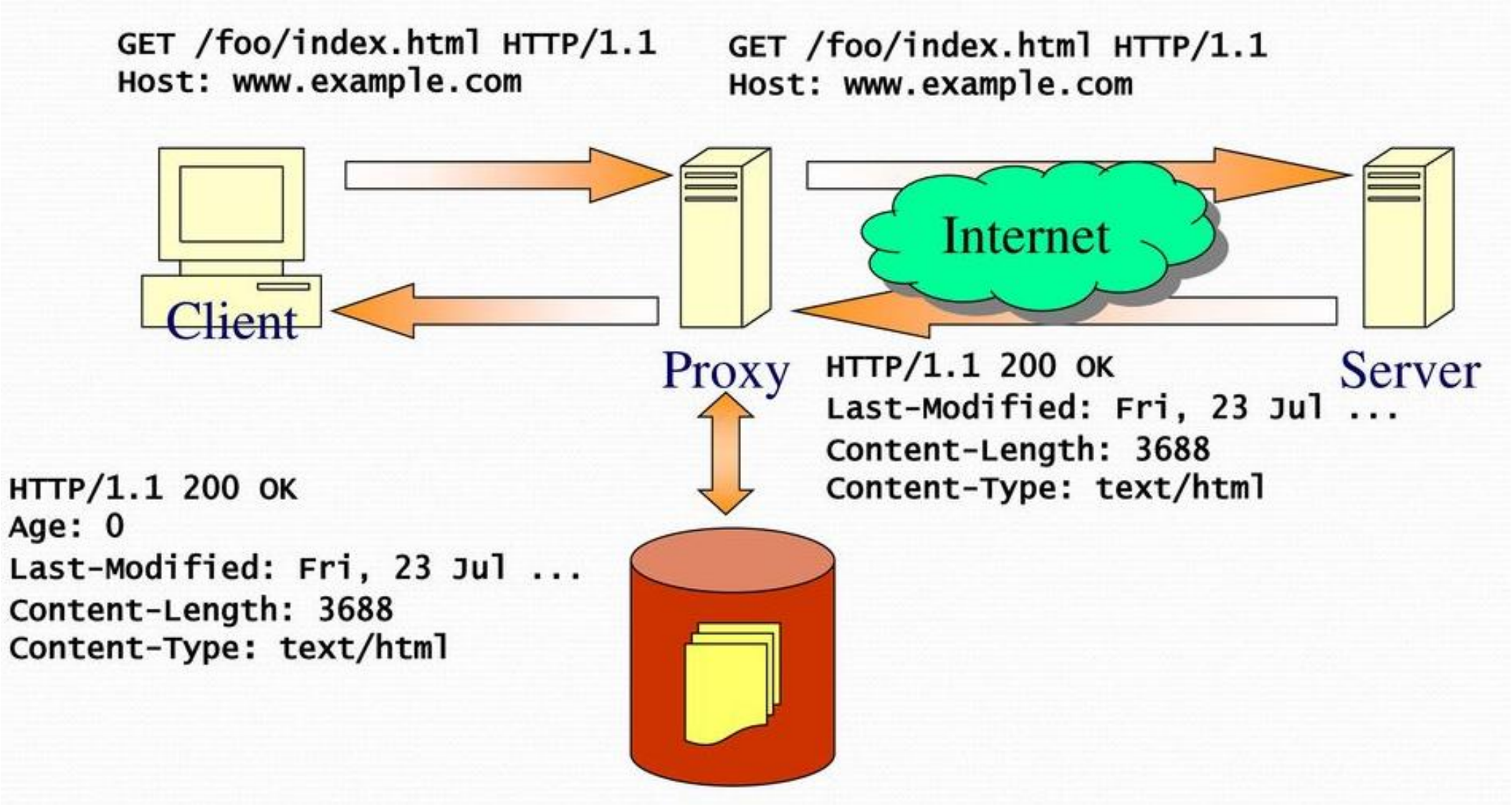
Example 4 – Revalidation on expiry (revalidate hit)



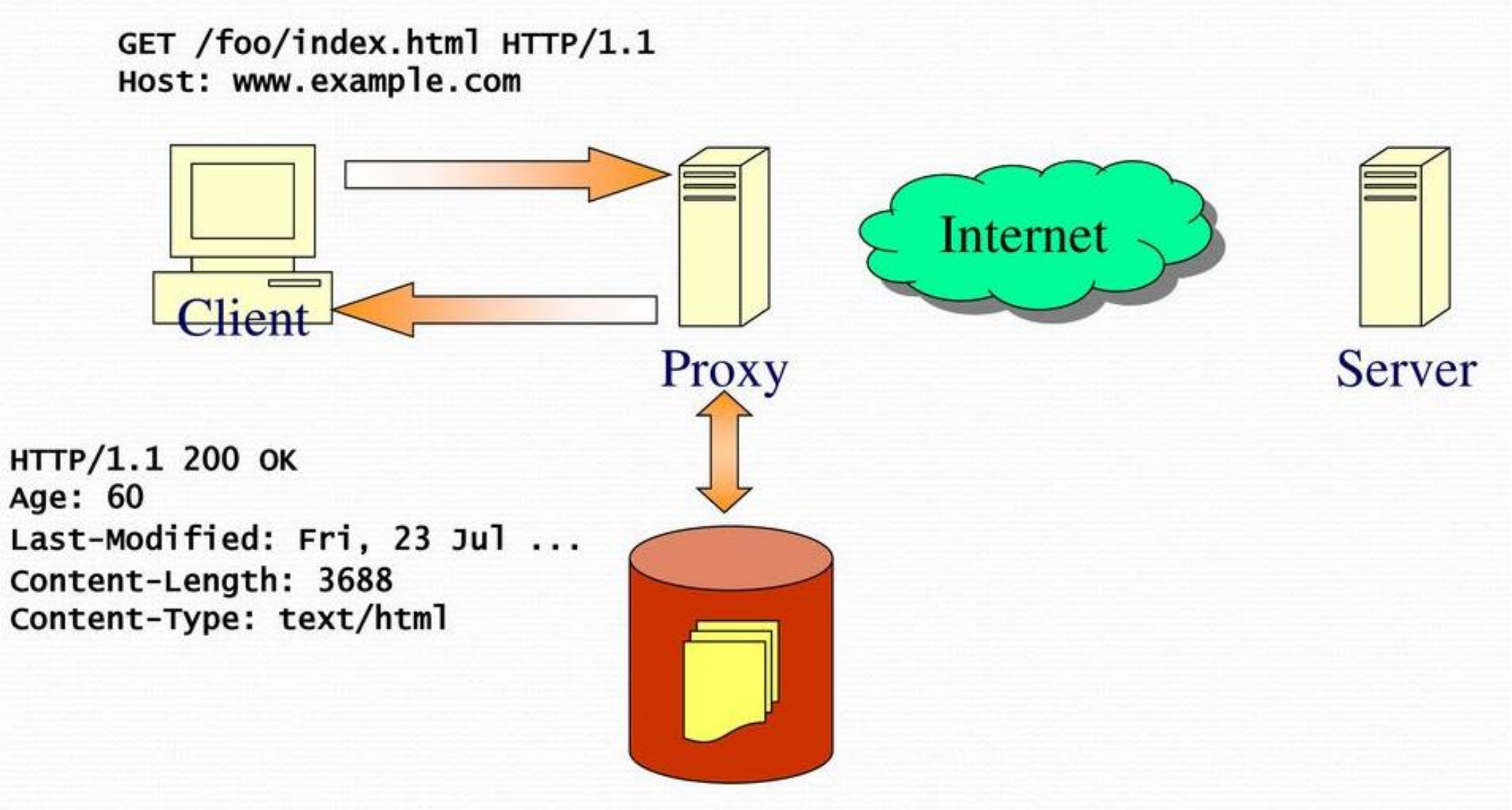
Example 5 – Revalidation on expiry (revalidate miss)



Example 6 – Proxy cache miss



Example 7 – Proxy cache hit



Client-side Caching - examples

Cache Control with mod_expires and mod_headers

For Apache/2.0, enable the modules in your httpd.conf file like this.

```
LoadModule expires_module modules/mod_expires.so
LoadModule headers_module modules/mod_headers.so
LoadModule deflate_module modules/mod_deflate.so
```

Client-side Caching – examples (Apache server)

Target Files by Extension for Caching

```
ExpiresActive On
...
<Directory "/home/website/public_html">
        Options FollowSymLinks MultiViews
        AllowOverride All
        Order allow,deny
        Allow from all
        ExpiresDefault A300
        <FilesMatch "\.html$">
                Expires A86400
        </FilesMatch>
        <FilesMatch "\.(gif|jpg|png|js|css)$">
                Expires A2592000
        </FilesMatch>
</Directory>
```

A300 sets the default expiry time to 300 seconds after access (A).

M300 set the expiry time to 300 seconds after file modification (M).

Client-side Caching – examples (Apache server)

Target Files by MIME Type

```
ExpiresActive On
ExpiresDefault "access plus 300 seconds"

<Directory "/home/website/public_html">
        Options FollowSymLinks MultiViews
        AllowOverride All
        Order allow,deny Allow from all
        ExpiresByType text/html "access plus 1 day“
        ExpiresByType text/css "access plus 1 day“
        ExpiresByType text/javascript "access plus 1 day“
        ExpiresByType image/gif "access plus 1 month"
        ExpiresByType image/jpg "access plus 1 month"
        ExpiresByType image/png "access plus 1 month"
        ExpiresByType application/x-shockwave-flash "access plus 1 day"
</Directory>
```

For expiry commands can be used **access** or **modified**

Client-side Caching – examples (IIS)

Output Caching and ASP.NET Core MVC

Cache profiles: Instead of duplicating response cache settings on many controller action attributes, cache profiles can be configured as options when setting up MVC/Razor Pages in Startup.ConfigureServices.

```
public void ConfigureServices(IServiceCollection services)
{
   services.AddRazorPages();
   services.AddMvc(options =>
   {
      options.CacheProfiles.Add("Default30",
         new CacheProfile()
         {
            Duration = 30
         });
   });
}
```

```
[ResponseCache(CacheProfileName = "Default30")]
public ActionResult CachedDateTime()
{
  ViewBag.Message = DateTime.Now.ToString();
  return View();
}
```

ASP.NET Core MVC

Client-side Caching – examples (IIS)

Properties

Duration	Gets or sets the duration in seconds for which the response is cached. If this property is set to a non null value, the "max-age" in "Cache-control" header is set in the Response.
Location	Gets or sets the location where the data from a particular URL must be cached. If this property is set to a non null value, the "Cache-control" header is set in the Response.
NoStore	Gets or sets the value which determines whether the data should be stored or not. When set to `true`, it sets "Cache-control" header in Response to "no-store". Ignores the "Location" parameter for values other than "None". Ignores the "Duration" parameter.
VaryByHeader	Gets or sets the value for the Vary header in Response.
VaryByQueryKeys	Gets or sets the query keys to vary by.

ASP.NET Core MVC

Client-side Caching – examples (IIS)

Vary

This header is only written when the VaryByHeader property is set. The property set to the `Vary` property's value. The following sample uses the VaryByHeader property:

```
[ResponseCache(VaryByHeader = "User-Agent", Duration = 30)]
public class Cache1Model : PageModel
{
```

```
Cache-Control: public,max-age=30
Vary: User-Agent
```

ASP.NET Core MVC

Client-side Caching – examples (IIS)

NoStore and Location.None

NoStore overrides most of the other properties. When this property is set to `true`, the `Cache-Control` header is set to `no-store`. If Location is set to `None`:

- `Cache-Control` is set to `no-store,no-cache`.
- `Pragma` is set to `no-cache`.

If NoStore is `false` and Location is `None`, `Cache-Control`, and `Pragma` are set to `no-cache`.

NoStore is typically set to `true` for error pages.

```
[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
public class Cache2Model : PageModel
{
```

```
Cache-Control: no-store,no-cache
Pragma: no-cache
```

ASP.NET Core MVC

Client-side Caching – examples (IIS)

Location and Duration

To enable caching, Duration must be set to a positive value and Location must be either `Any` (the default) or `Client`. The framework sets the `Cache-Control` header to the location value followed by the `max-age` of the response.

Location's options of `Any` and `Client` translate into `Cache-Control` header values of `public` and `private`, respectively. As noted in the NoStore and Location.None section, setting Location to `None` sets both `Cache-Control` and `Pragma` headers to `no-cache`.

`Location.Any` (`Cache-Control` set to `public`) indicates that the *client or any intermediate proxy* may cache the value, including Response Caching Middleware.

`Location.Client` (`Cache-Control` set to `private`) indicates that *only the client* may cache the value. No intermediate cache should cache the value, including Response Caching Middleware.

ASP.NET Core MVC

Client-side Caching – examples (IIS)

Location and Duration

```
[ResponseCache(Duration = 10, Location = ResponseCacheLocation.Any, NoStore = false)]
public class Cache3Model : PageModel
{
```

```
Cache-Control: public,max-age=10
```

HTTP Authentication

HTTP Authentication is a security mechanism to verify the user who is eligible to access the web resource.

It involves communication between client and server using HTTP header that represents server requesting user's credentials for authentication. The client in response provides the information in the header.

Authorization

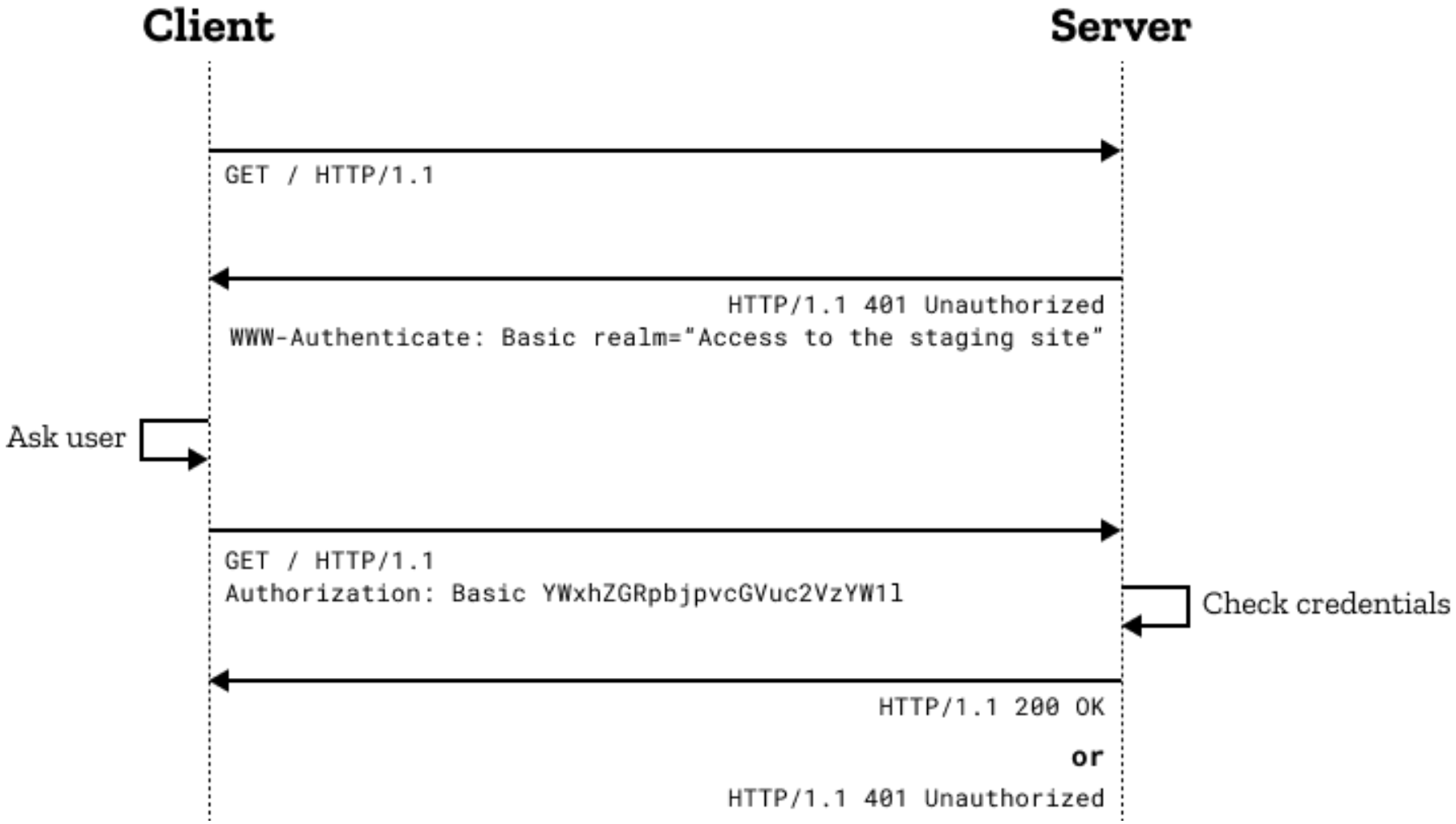
The **WWW-Authenticate** header is sent along with a **401 Unauthorized** response.

WWW-Authenticate: <type> realm=<realm>

<type>
Authentication type (HTTP Authentication Scheme). A common type is "Basic".

realm=<realm>
A description of the protected area. If no realm is specified, clients often display a formatted hostname instead.

Authorization



Authorization

HTTP Authentication Schemes

- **Basic** [RFC7617]
- **Bearer** [RFC6750]
- **Digest** [RFC7616]
- **HOBA** [RFC7486, Section 3] The HOBA scheme can be used with either HTTP servers or proxies. When used in response to a 407 Proxy Authentication Required indication, the appropriate proxy authentication header fields are used instead, as with any other HTTP authentication scheme.
- **Mutual** [RFC8120]
- **Negotiate** [RFC4559, Section 3] This authentication scheme violates both HTTP semantics (being connection-oriented) and syntax (use of syntax incompatible with the WWW-Authenticate and Authorization header field syntax).
- **OAuth** [RFC5849, Section 3.5.1]
- **SCRAM-SHA-1** [RFC7804]
- **SCRAM-SHA-256**[RFC7804]
- **vapid** [RFC-ietf-webpush-vapid-04, Section 3]

Web Cookies… what are they?

Cookies

Session

Persistent

A ***session cookie*** exists only in temporary memory while the user navigates a website. Session cookies expire or are deleted when the user closes the web browser. Session cookies are identified by the browser by the absence of an expiration date assigned to them.

Persistent cookies are stored on a user's device to help remember information, settings, preferences, or sign-on credentials that a user has previously saved. These cookies have an expiration date issued to it by the webserver. Basically, this type of cookie is saved on your computer so when you close it and start it up again, the cookie is still there. Once the expiration date is reached, it is destroyed by the owner.

Cookies

First and Third-Party Cookies ?

First-party cookies are stored by the domain (website) you are visiting directly. They allow website owners to collect analytics data, remember language settings, and perform other useful functions that help provide a good user experience.

Third-party cookies are created by domains other than the one you are visiting directly, hence the name third-party. They are used for cross-site tracking, retargeting and ad-serving.

Cookies

Malicious Cookies?
(Tracking Cookies)

What Are Tracking Cookies?

Tracking cookies are a type of internet cookie used primarily for advertising purposes.
As a user surfs the web, the cookies follow them, keeping track of information about the user's preferences, habits, past website visits, and purchases.
With this information, you can send advertisements to the user and show them the products and services they're most likely interested in — among other actions.

Tracking cookies

What Data Do Tracking Cookies Store?

Common pieces of personal information collected include:

- Type of device the user used (e.g., computer, tablet, mobile phone)
- Name and age
- Website preferences, themes, and settings (language, notifications, time zone)
- IP address
- Email address and passwords
- History and prior purchases
- Time spent on webpages
- Browsing history
- Websites visited
- Advertisement interactions and clicks
- Search engine inputs

Cookies

```
GET /index.html HTTP/1.1
Host: www.example.org
```

browser --------→ server

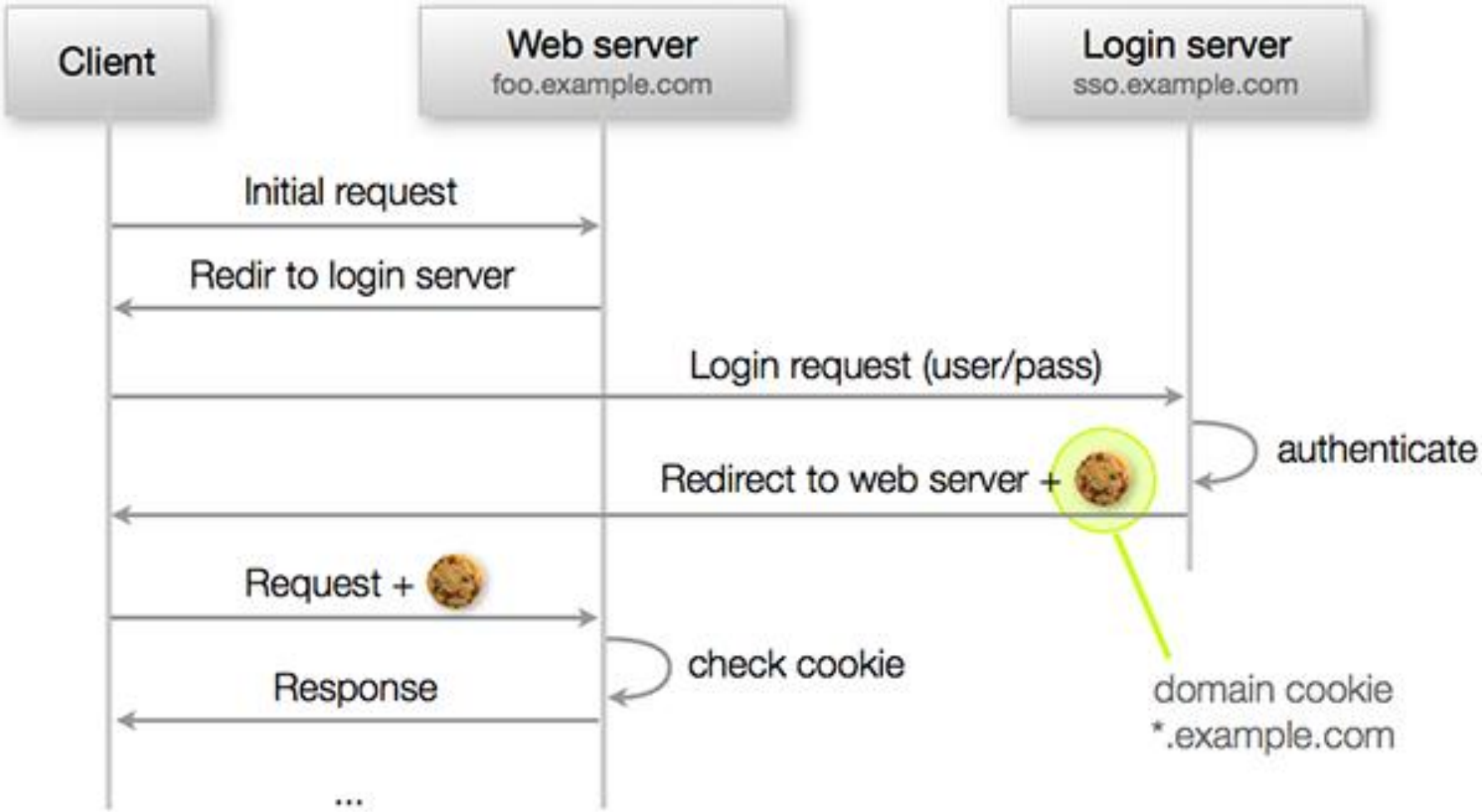
```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT

(content of page)
```

browser ←-------- server

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
Accept: */*
```

browser --------→ server

Cookies – Authentication example



Cookies

```
Cookie: key1=value1; key2=value2; ...
```

Structure

- Name
- Value
- Date to expire
- Path where it applies
- Domain where it applies
- Need secure connection
- Can be accessed by means other than HTTP (javascript, etc.)

Cookies

Name

Path

Just for
HTTP messages
(cannot be accessed by
scripts)

```
Set-Cookie: LSID=DQAAAK…Eaem_vYg; Domain=docs.foo.com; Path=/accounts; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
Set-Cookie: HSID=AYQEVn….DKrdst; Domain=.foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; HttpOnly
Set-Cookie: SSID=Ap4P….GTEq; Domain=.foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
 ......
```

Domain

Validity

Secure connection?

Cookies are a problem?

How to use sessions without cookies?

cookieless

querystring

url part

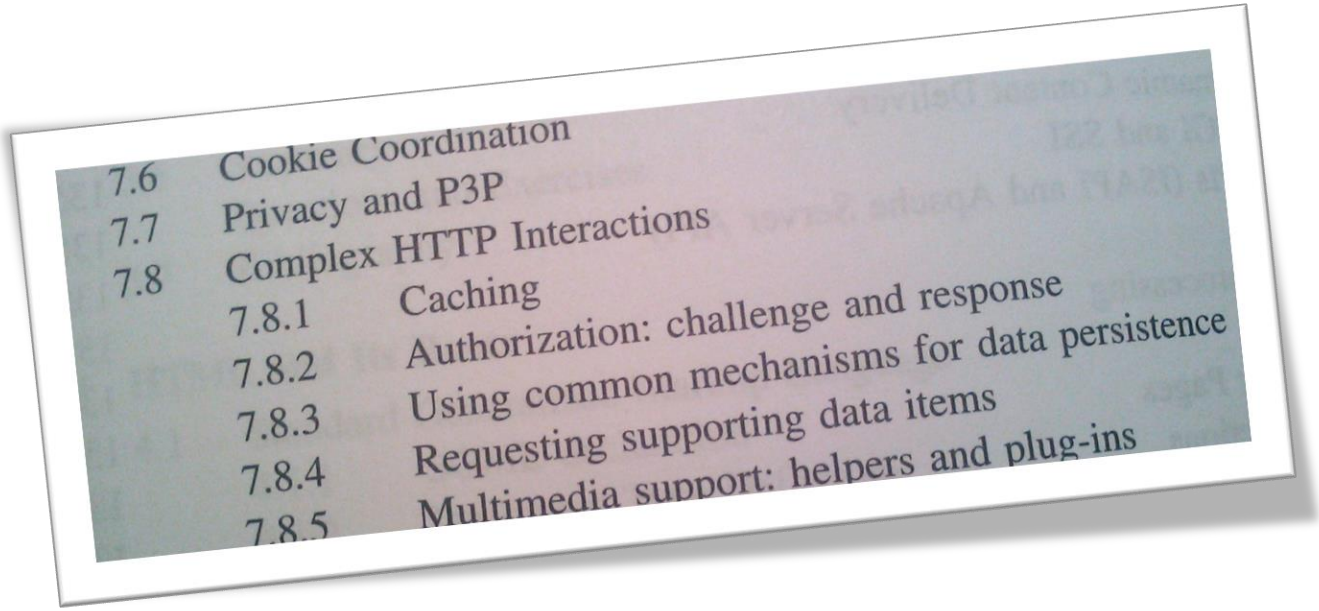
http://www.somewebsite.com/page.aspx?sid=jrkwojeqrojq3op349023231234r23rf2

http://yourserver/folder/(session ID here)/default.aspx

Bibliography

Shklar, Leon & Rosen, Rich (2009). *Web Application Architecture: Principles, Protocols and Pratices*. Chichester, Reino Unido: John Wiley & Sons.

Pages: 172 to 182



7.6 Cookie Coordination
7.7 Privacy and P3P
7.8 Complex HTTP Interactions
7.8.1 Caching
7.8.2 Authorization: challenge and response
7.8.3 Using common mechanisms for data persistence
7.8.4 Requesting supporting data items
7.8.5 Multimedia support: helpers and plug-ins

Resume

COMPLEX HTTP INTERACTIONS

WEB CACHE:

CONCEPT, HTTP HEADERS INVOLVED, AGENTS BEHAVIOR (BROWSER, SERVER, PROXIES)

HTTP AUTHORIZATION:

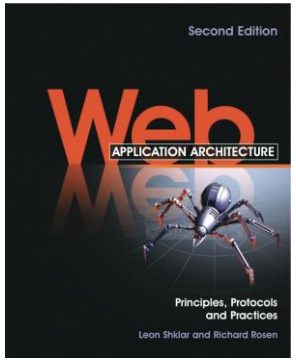
CONCEPT, HTTP HEADERS INVOLVED, AGENTS BEHAVIOR (BROWSER, SERVER, PROXIES)

WEB COOKIES:

CONCEPT, TYPES, STRUCTURE, HTTP HEADERS INVOLVED, AGENTS BEHAVIOR (BROWSER, SERVER, PROXIES)

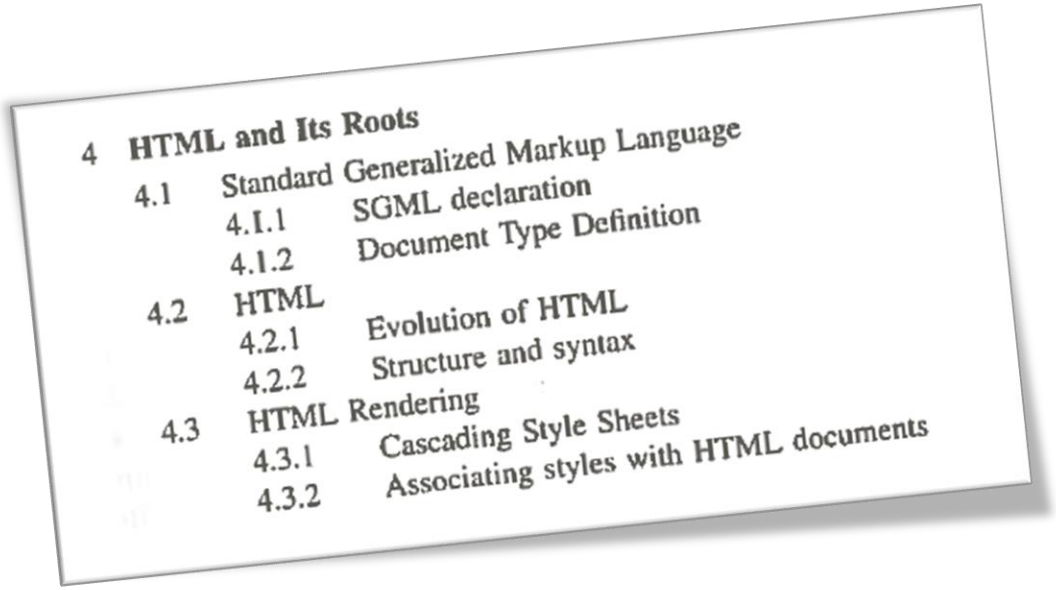
Next class

HTML and its origins



Shklar, Leon & Rosen, Rich (2009). *Web Application Architecture: Principles, Protocols and Pratices*. Chichester, Reino Unido: John Wiley & Sons.

Pages: 63 to 83



4 **HTML and Its Roots**

4.1 Standard Generalized Markup Language
- 4.1.1 SGML declaration
- 4.1.2 Document Type Definition

4.2 HTML
- 4.2.1 Evolution of HTML
- 4.2.2 Structure and syntax

4.3 HTML Rendering
- 4.3.1 Cascading Style Sheets
- 4.3.2 Associating styles with HTML documents