HTTP transactions: headers

- ...are text lines providing (meta-)information about:
  - the client;
  - the server;
  - the body of the HTTP message (which is called the entity).

- Some of their uses:
  - handling multimedia content;
  - internationalization;
  - redirection;
  - logging;
  - session maintenance;
  - …

HTTP transactions: header lines in action

Example: retrieving a cached document.

```
GET / HTTP/1.1
If-Modified-Since: Fri, 03 May 2013 16:01:18 GMT
```

Response:

```
HTTP/1.1 304 Not Modified
Date: Wed, 14 May 2013 22:17:02 GMT
Server: Apache/2.0.43 (Win32)
Connection: close
ETag: "1f164-5d6-8ba74f80;1f180-9c0-f1868a00"
Content-Location: index.html.en
Last-Modified: Thu, 02 May 2013 04:58:08 GMT
Expires: Wed, 22 May 2013 22:17:02 GMT
...
```

HTTP transactions: their general structure

Request:

Method + URL line
General headers
Request headers
Entity headers
Entity body

Response:

Status line
General headers
Response headers
Entity headers
Entity body

HTTP transactions: general header lines

- **`Cache-Control`**: specifies behavior for caching.
- **`Connection`**: indicates whether or not the network connection should close after this exchange.
- **`Date`**: specifies the current date.
- **`MIME-Version`**: specifies the version of MIME used in the HTTP transaction.
- **`Transfer-Encoding`**: indicates the type of transformation that was applied to the message body for safe transfer.
- **`Via`**: indicates the intermediate protocols and hosts that processed this HTTP transaction (used by gateways and proxies).
- …

HTTP transactions: request header lines

- **`Accept`**: specifies which media formats the client accepts.
- **`Cookie`**: used to convey data stored for the specific URL (more below).
- **`Host`**: specifies the host and port number that the client connected to (required for all clients in HTTP/1.1).
- **`If-Modified-Since`**: requests the document only if newer than the specified date.
- **`Referer`**: (*sic!*) specifies the URL of the previous document that contained the link to this one.
- **`User-Agent`**: identifies the client program.
- ...

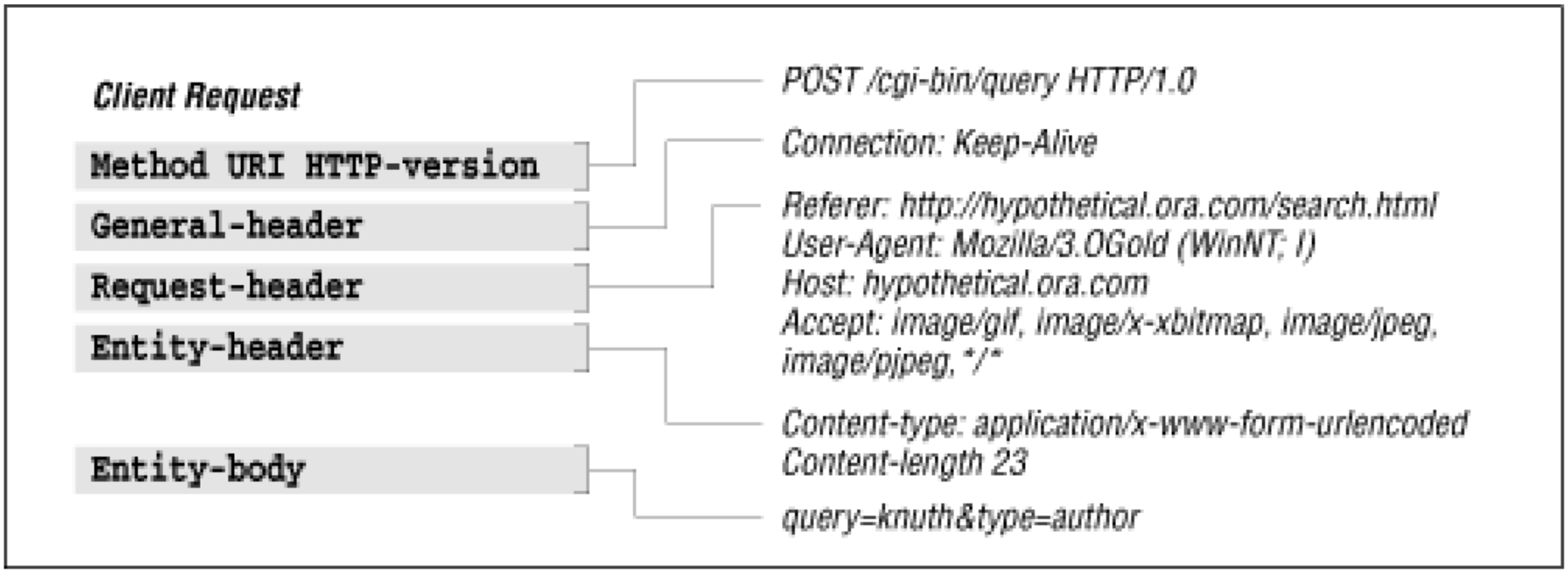
HTTP transactions: response header lines

- **`Retry-After`**: specifies either the number of seconds or a date after which the server becomes available again.
- **`Server`**: specifies the name and version number of the server.
- **`Set-Cookie`**: defines a *name*=*value* pair to be associated with this (server and) URL (more below).
- **`WWW-Authenticate`**: specifies an authorization type and realm of authorization.
- ...

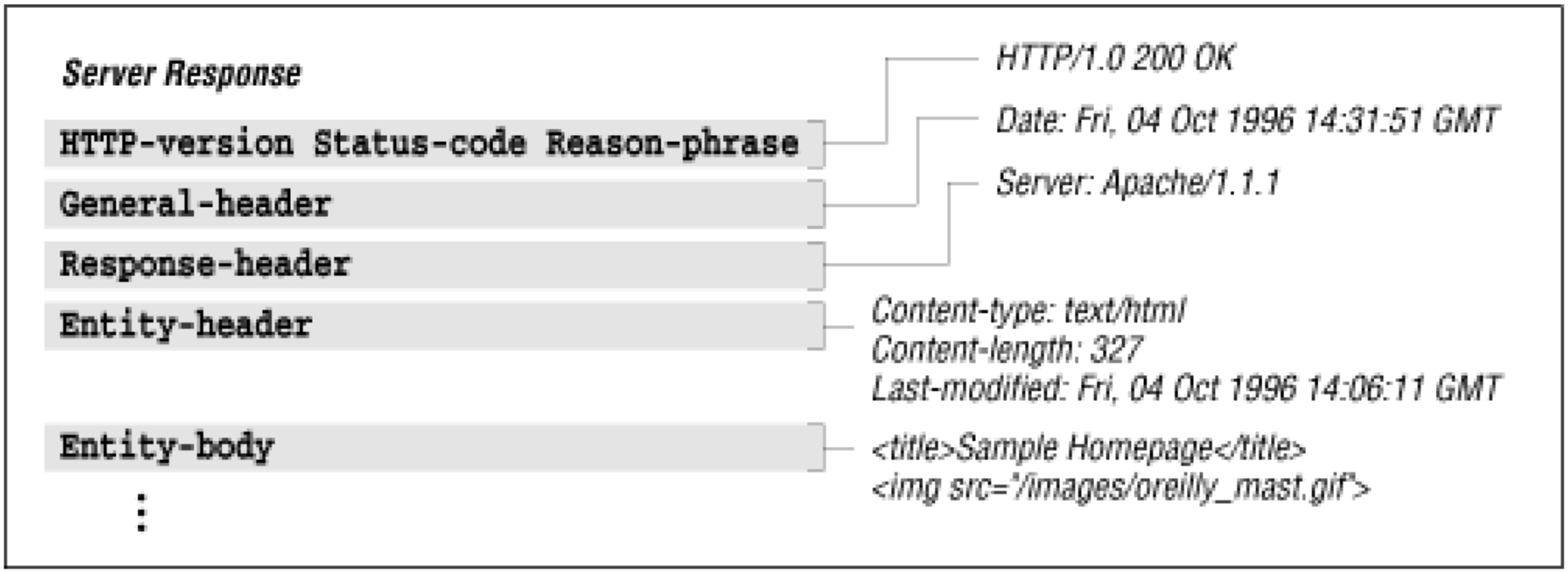
HTTP transactions: entity header lines

- **`Content-Language`**: specifies the natural language used in the document (entity) being returned.
- **`Content-Length`**: specifies the length of the entity.
- **`Content-Type`**: specifies the MIME type of the entity.
- **`Expires`**: gives a date and time when contents may change.
- **`Last-Modified`**: gives the date and time that the entity was last changed.
- **`Location`**: specifies the location of a created or moved document.
- ...

HTTP transactions: example client request



HTTP transactions: example server response



Hypermedia: MIME

- MIME: Multipurpose Internet Mail Extensions.

- Originally developed for sending non-text email attachments.

- MIME is used from HTTP/1.0 to enable sending "hypermedia" instead of simply hypertext.
  - For example: an image file, or sound file.

- Processing is based on MIME types.
- Used in HTTP headers, e.g. `Content-type: text/html`.

- Used by server to indicate type of media (e.g. may associate filename extensions of resources with MIME types).
- Then used by client to interpret the response body (or let a plugin or external program handle it...).

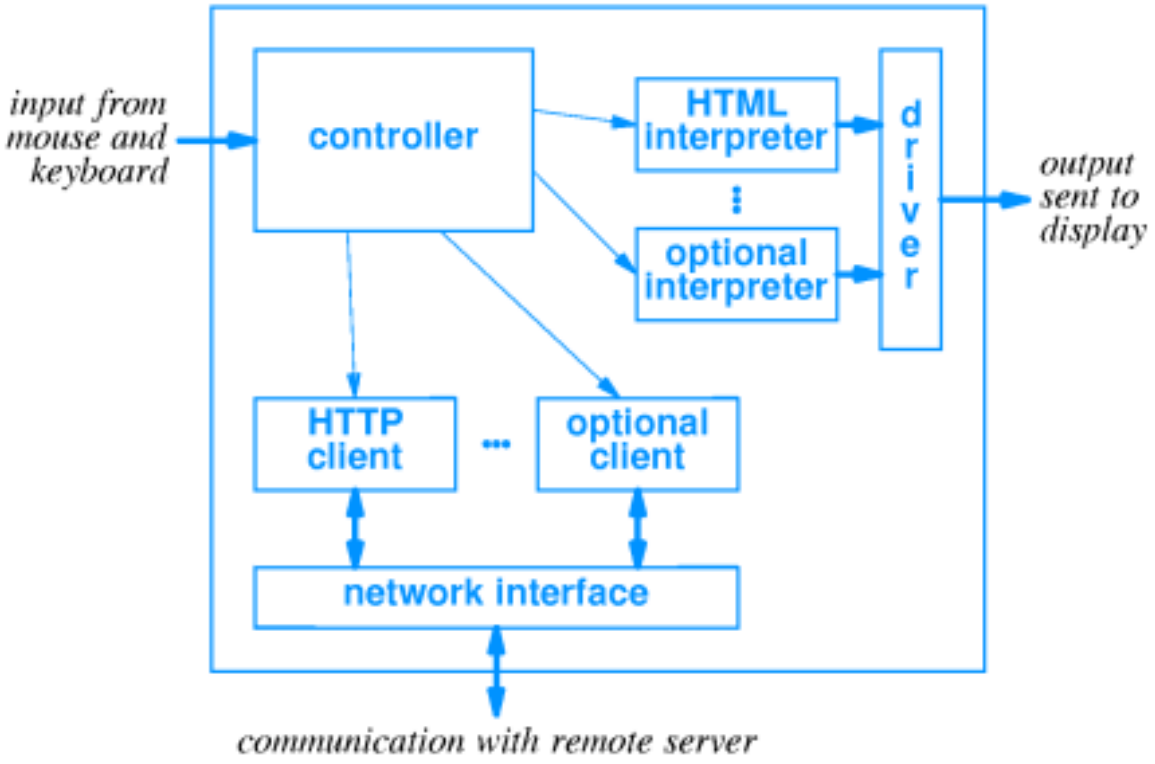
Hypermedia: examples of MIME types

- text/plain
- text/html
- text/xml
- image/gif
- image/jpeg
- audio/wav
- video/mpeg
- application/msword

- ...See http://www.iana.org/assignments/media-types/.

The browser: basic functionality

- The hypermedia retrieve-and-display machine →

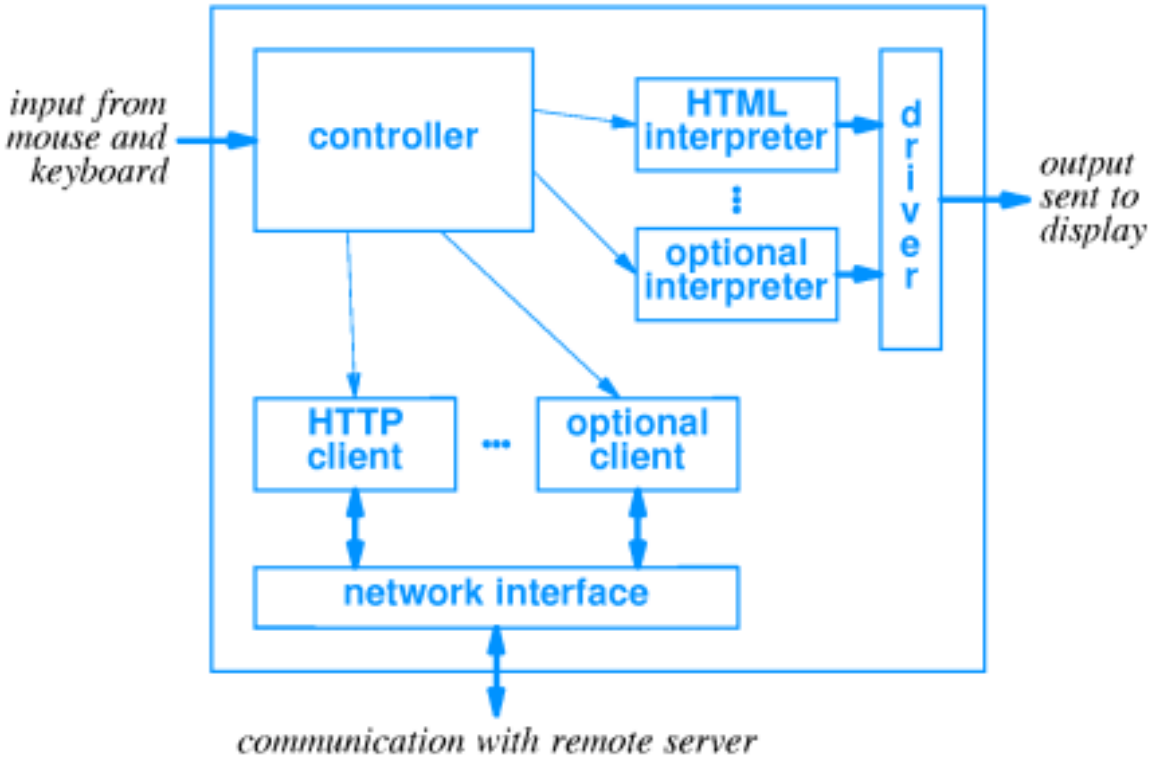


- *Web browser:* an HTTP client that can retrieve, display and offer navigation of HTML hypertext documents.

- *Web server:* an HTTP server that can return HTML hypertext documents on request.

The browser: enabling extensible hypermedia

- The hypermedia retrieve-and-display machine →



- Optional interpreter can be: *JavaScript interpreter, Flash plugin, XML parser, …*
- Optional client can be: *FTP client, news client, mail client, …*

The browser: underlying hardware

- Browsers run on personal computing devices.

- *Over the years:* a shift in form factors.

  Let's take a look at recent data from StatCounter >

Taking stock: The World Wide Web (WWW)

- The infrastructure of web browsers and web servers implements the *World Wide Web (WWW)*.

- The WWW: a graph of *hypertext documents*, on top of a graph of internetnetworked computers.

- **One definition of the World Wide Web:**
  all the resources and users on the Internet
  that are using the Hypertext Transfer Protocol (HTTP).

- (The web is "just" *one* way of using the Internet!)

Taking stock: The World Wide Web (WWW)

- Begun in 1989 by Tim Berners-Lee at CERN.
- TBL created HTTP, URLs, HTML (sort of), and the first web-server and web-browser as we know it (1990).
- In 1991, the WWW was available on the Internet.
- TBL became director of W3C, the organization that 'manages' the World Wide Web, and a researcher at MIT.
- He never made any (commercial) profits from his invention.

Taking stock: What next?

- We have obtained structured, navigable hypertext.

- This enables quick traversal of a graph of hypertext documents.

- This is an old dream (“*the ultimate library!*”) come true
  – but once obtained, new dreams appear.

- Not only do we want to ask for existing hypertext documents by their names...

- ...we also want new documents to be created *in the moment,* with *contents according to our wishes*.

- ↑ Think of what a search engine does (and then, think of doing this using only clickable links).

Interactive hypertext: HTML forms

- When surfing the web, you request web pages *from* servers.
- How, while doing this, do you submit information *to* servers?
- One way is via HTML *forms*.
- These provide interactivity between web user and web application.
- Originally used to provide data to "CGI scripts".

Interactive hypertext: example of an HTML form

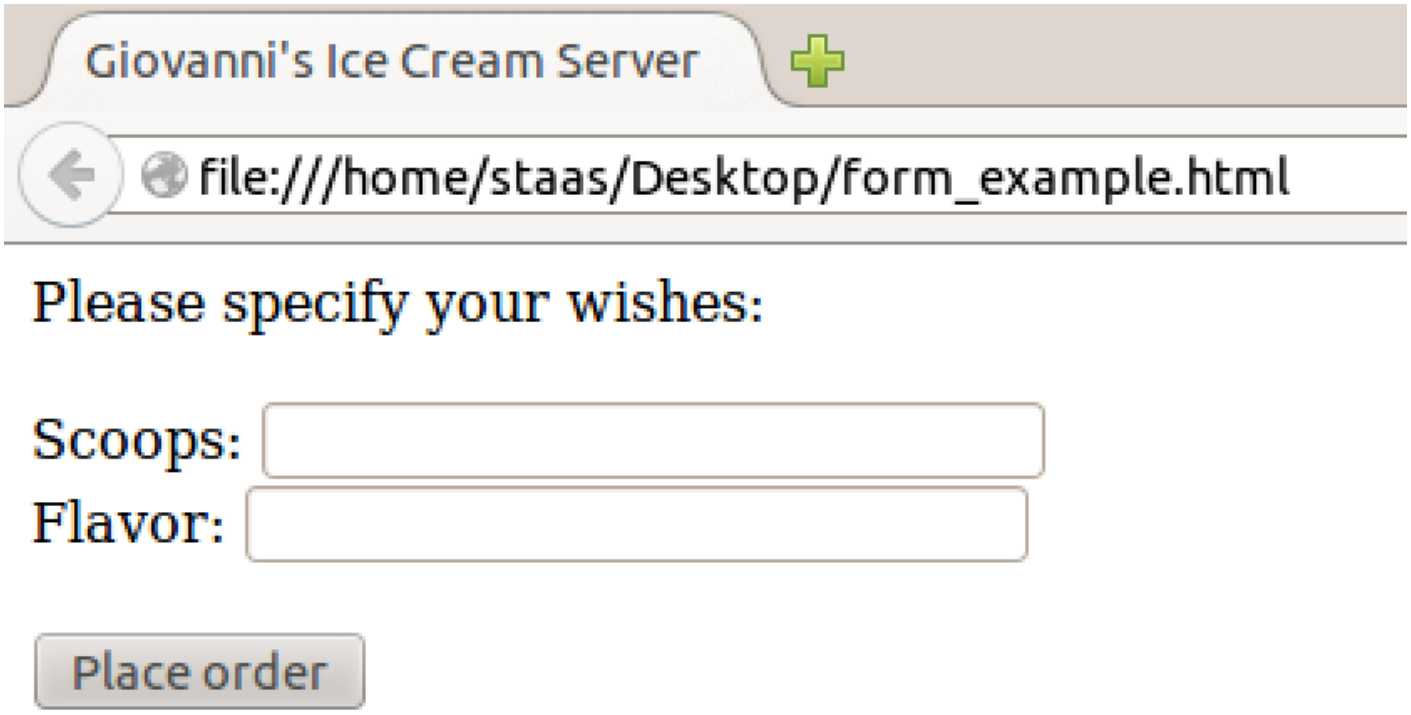
```
<html>
  <head> <title> Giovanni's Ice Cream Server </title> </head>

  <body>

    <form method="post" action="/cgi-bin/ice.cgi">
      Please specify your wishes: <br> <br>
      Scoops: <input type="text" name="scoops" maxlength="1"> <br>
      Flavor: <input type="text" name="flavor"> <br> <br>
      <input type="submit" value="Place order">
    </form>

  </body>
</html>
```

Interactive hypertext: example of an HTML form



URL encoding of HTML form data

- The web client encodes the data entered into an HTML form before passing it to the web server. It:

- Collects all form element names and values;
- glues each name, value pair together with '**=**';
- glues resulting pairs together with '&';
- replaces space characters by '**+**';
- replaces all non-alphanumeric characters by '**%xx**', where `xx` is their hexadecimal ASCII code.

- Example: *Amélie Poulain* entered *2* scoops of *cherry* ice cream:

```
customer=Am%E9lie+Poulain&scoops=2&flavor=cherry
```

...under the hood, then carried by HTTP

Next, the URL-encoded form data is transferred from client to server using either:

- a GET request, with the encoded data in the URL query string, after a '?' character:

```
GET /cgi-bin/ice.cgi?scoops=2&flavor=cherry HTTP/1.0
```

- a POST request, with the encoded data in the entity body:

```
POST /cgi-bin/ice.cgi HTTP/1.0
Content-type: application/x-www-form-urlencoded
Content-length: 22

scoops=2&flavor=cherry
```

Interactive hypertext: HTTP sessions

- In principle, HTTP is a stateless protocol:
  an HTTP response depends on its corresponding request *only*.
  Not on previous requests from that same client.

- After a client-server HTTP transaction is completed, the TCP connection is terminated.
- On a later request, the client must establish a new connection.

- For *prolonged* interactivity, the server must become able to relate a series of multiple client HTTP requests over time, into a session.

- Example: filling a virtual shopping cart on an e-commerce website.

- Implementing a session requires a mechanism to relate separate client requests by maintaining a session ID throughout HTTP communication.

- Three commonly used mechanisms exist:

Interactive hypertext: HTTP sessions

- Implementing a session requires a mechanism to relate separate client requests by maintaining a session ID throughout HTTP communication.

- Three commonly used mechanisms exist:

  - *Cookies*: include the session ID as a text value in HTTP request and response header fields. The server maintains a list of cookie values to implement state between multiple HTTP transactions.

  - *URL rewriting*: each URL linking to the next page is rewritten to include the session ID in the query string (after the '?' character) *www.liacs.nl/hello.index?id=19680128* .

  - *Hidden fields*: include a form in each page with a hidden field containing the value of the session ID (user is unaware of the form).

Under the hood: HTTP cookies

- An HTTP cookie is a unit of textual information
  sent by a web server for storage on a client machine, and
  sent back by the client each time it accesses that same server:

- An HTTP server uses the **`Set-cookie:`** *response header field*:

```
Set-cookie: cart=2242; items=198+234; expire: Wed,…
Set-cookie: sex=male; car=1991-Peugeot-205-1.1
```

- When the client requests another page from the same server,
  the cookie is returned via the **`Cookie:`** *request header field*:

```
Cookie: cart=2242; items=198+234
Cookie: sex=male; car=1991-Peugeot-205-1.1
```

HTTP cookies and privacy

- Cookies are as widely useful as the HTTP sessions they enable.
- But this also means they can be used to compromise a user's privacy!
- The browsing actions of a user can be tracked over time and stored, without the user being made aware of this.
- In the Netherlands, legislation against this has been introduced.

HTTP cookies and privacy

- Example of tracking using cookies:

  1) While the user views some webpage *a.html* on server *www.seems-innocent.com*...

  ...images embedded in *a.html* from server *www.spamlord.com* are loaded automatically, without the user being aware.

  2) The resulting HTTP requests to *www.spamlord.com* are made to include a **cookie**, uniquely identifying the user to that server.

  3) Now, when other webpages *b.html*, *c.html*, ... are visited later, *www.spamlord.com* can use similar images to identify the user again, log web-surfing habits, ban him/her, serve custom advertisements, etc.

HTTP cookies and privacy

- Let's have a look at some tools for “countertracking”:
  - Mozilla Lightbeam
  - Ghostery

>