Lezione 16 - 09 04 17

Monday, April 9, 2018 12:25

10.12 Pragma

The Pragma general-header field is used to include implementation-specific directives that may apply to any recipient along the request/response chain. All pragma directives specify optional behavior from the viewpoint of the protocol; however, some systems may require that behavior be consistent with the directives.

```
Pragma            = "Pragma" ":" 1#pragma-directive

pragma-directive = "no-cache" | extension-pragma
extension-pragma = token [ "=" word ]
```

10.9 If-Modified-Since: quando se la copia su file è aggiornata

A conditional GET method requests that the identified resource be transferred only if it has been modified since the date given by the If-Modified-Since header. The algorithm for determining this includes the following cases:

a) If the request would normally result in anything other than a 200 (ok) status, or if the passed If-Modified-Since date is invalid, the response is exactly the same as for a normal GET. A date which is later than the server's current time is invalid.

b) If the resource has been modified since the If-Modified-Since date, the response is exactly the same as for a normal GET.

c) If the resource has not been modified since a valid If-Modified-Since date, the server shall return a 304 (not modified) response.

→ questo controllo lo fa il server

The purpose of this feature is to allow efficient updates of cached information with a minimum amount of transaction overhead.

È una GET condizionale, la fa solo se → If-Modified-Since header (request)

Questo controllo può farlo anche il client guardando la data di scadenza del file → entity header

10.10 Last-Modified

Metodo HAD (al posto di GET) → risposta solo l'header senza entity body

La scadenza me la dice

↳ 10.37 Expires

Ci dà la data in cui si deve considerare il file scaduto

SESSIONI:

10.16 WWW - Authenticate

The WWW-Authenticate response-header field must be included in 401 (unauthorized) response messages. The field value consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable to the Request-URI.

Accesso al server tramite autenticazione

```
WWW-Authenticate = "WWW-Authenticate" ":" 1#challenge
```

→ metodo di autenticazione

es. password

The HTTP access authentication process is described in Section 11. User agents must take special care in parsing the WWW-Authenticate field value if it contains more than one challenge, or if more than one WWW-Authenticate header field is provided, since the contents of a challenge may itself contain a comma-separated list of authentication parameters.

Non è un metodo di sicurezza, viaggia tutto in chiaro

↳ bisogna cifrare il canale

11. Access Authentication

HTTP provides a simple challenge-response authentication mechanism which may be used by a server to challenge a client request and by a client to provide authentication information. It uses an extensible, case-insensitive token to identify the authentication scheme, followed by a comma-separated list of attribute-value pairs which carry the parameters necessary for achieving authentication via that scheme.

11.1

```
auth-scheme    = token
auth-param     = token "=" quoted-string
```

es. "Basic" per Basic Authentication Scheme

Sinonimo di dominio)

Realm: dominio utenti che accede (stringa)

es. Studenti unipd

The 401 (unauthorized) response message is used by an origin server to challenge the authorization of a user agent. This response must include a WWW-Authenticate header field containing at least one challenge applicable to the requested resource.

```
challenge      = auth-scheme 1*SP realm *( "," auth-param )

realm          = "realm" "=" realm-value
realm-value    = quoted-string
```

```
userid-password    = [ token ] ":" *TEXT
```

If the user agent wishes to send the user-ID "Aladdin" and password "open sesame", it would use the following header field:

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

```
basic-credentials = "Basic" SP basic-cookie

basic-cookie      = <base64 [5] encoding of userid-password,
                     except not limited to 76 char/line>
```

andrebbe a capo la codifica standard, ma dentro un header non si può

To receive authorization, the client sends the user-ID and password, separated by a single colon (":") character, within a base64 [5] encoded string in the credentials.

base64 → RFC 1521

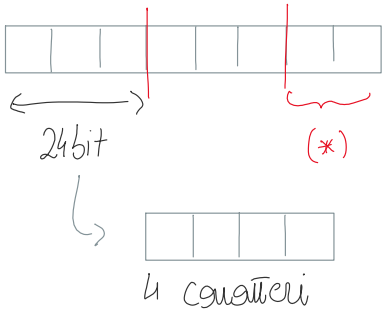
Mime (Multipurpose Internet Mailing Extensions)

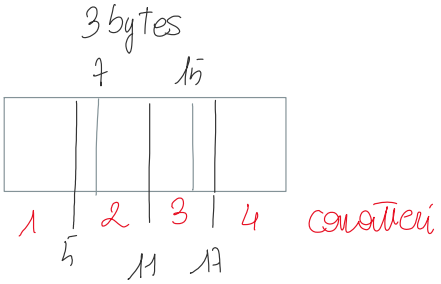
5.2 Base 64 Content-Transfer-Encoding

24 bit = 8 x 3 bit caratteri = 6 x 4 bit caratteri → in ASCII

10 numeri
26 lettere maiuscole
26 lettere minuscole
"+"
"/"

$2^6 = 64$ caratteri dell'alfabeto





(*) mi invento un carattere fantasma fatto di tutti zeri

Così facendo l'ultimo carattere che non serve viene codificato con un "="

Il penultimo carattere ha 2 bit forzati a zero per convenzione

Può succede di avere anche un solo byte alla fine → devo due caratteri fantasma

Avrò due "=" e solo i primi due bit del secondo carattere sono bit veri.

Esercizio: implementare la codifica Base 64 vedi man base64

```
bedlia@localhost:~$ base64
ciao step
Y2lhbyBzdGVwCg==
bedlia@localhost:~$ ps CTRL + "D" per terminare l'input step
```

← ps. mi ha contato anche l'invio

Ho solo due casi (e non 24) perché si trasmette a byte e non a bit.

Però non si vuole che l'autenticazione avvenga ogni volta se è già

Stato sessione → poiché l'http è stateless, il server non può ricordarsi

⇒ lo fa il client

appena il server ci manda un cookie da mettere nella richiesta, che ci identifica ⇒ non è un'autenticazione quindi va bene → RFC 6265 sezione 4 e 5