cachecontrol: HTTP Caching Parser and Interpretation

GO reference build error

`cachecontrol` implements RFC 7234 **Hypertext Transfer Protocol (HTTP/1.1): Caching**. It does this by parsing the `Cache-Control` and other headers, providing information about requests and responses -- but `cachecontrol` does not implement an actual cache backend, just the control plane to make decisions about if a particular response is cachable.

Usage

`cachecontrol.CachableResponse` returns an array of reasons why a response should not be cached and when it expires. In the case that `len(reasons) == 0`, the response is cachable according to the RFC. However, some people want non-compliant caches for various business use cases, so each reason is specifically named, so if your cache wants to cache `POST` requests, it can easily do that, but still be RFC compliant in other situations.

Examples

Can you cache Example.com?

```go
package main

import (
    "github.com/pquerna/cachecontrol"

    "fmt"
    "io/ioutil"
    "net/http"
)

func main() {
    req, _ := http.NewRequest("GET", "http://www.example.com/", nil)

    res, _ := http.DefaultClient.Do(req)
    _, _ = ioutil.ReadAll(res.Body)

    reasons, expires, _ := cachecontrol.CachableResponse(req, res,
cachecontrol.Options{})

    fmt.Println("Reasons to not cache: ", reasons)
    fmt.Println("Expiration: ", expires.String())
}
```

Can I use this in a high performance caching server?

`cachecontrol` is divided into two packages: `cachecontrol` with a high level API, and a lower level `cacheobject` package. Use [Object](#) in a high performance use case where you have previously parsed headers containing dates or would like to avoid memory allocations.

```
package main

import (
    "github.com/pquerna/cachecontrol/cacheobject"

    "fmt"
    "io/ioutil"
    "net/http"
)

func main() {
    req, _ := http.NewRequest("GET", "http://www.example.com/", nil)

    res, _ := http.DefaultClient.Do(req)
    _, _ = ioutil.ReadAll(res.Body)

    reqDir, _ := cacheobject.ParseRequestCacheControl(req.Header.Get("Cache-
Control"))

    resDir, _ := cacheobject.ParseResponseCacheControl(res.Header.Get("Cache-
Control"))
    expiresHeader, _ := http.ParseTime(res.Header.Get("Expires"))
    dateHeader, _ := http.ParseTime(res.Header.Get("Date"))
    lastModifiedHeader, _ := http.ParseTime(res.Header.Get("Last-Modified"))

    obj := cacheobject.Object{
        RespDirectives:         resDir,
        RespHeaders:            res.Header,
        RespStatusCode:         res.StatusCode,
        RespExpiresHeader:      expiresHeader,
        RespDateHeader:         dateHeader,
        RespLastModifiedHeader: lastModifiedHeader,

        ReqDirectives: reqDir,
        ReqHeaders:    req.Header,
        ReqMethod:     req.Method,

        NowUTC: time.Now().UTC(),
    }
    rv := cacheobject.ObjectResults{}

    cacheobject.CachableObject(&obj, &rv)
    cacheobject.ExpirationObject(&obj, &rv)

    fmt.Println("Errors: ", rv.OutErr)
    fmt.Println("Reasons to not cache: ", rv.OutReasons)
    fmt.Println("Warning headers to add: ", rv.OutWarnings)
    fmt.Println("Expiration: ", rv.OutExpirationTime.String())
}
```

Improvements, bugs, adding features, and taking cachecontrol new directions!

Please open issues in Github for ideas, bugs, and general thoughts. Pull requests are of course preferred :)

Credits

`cachecontrol` has recieved significant contributions from:

- Paul Querna

License