# Good Design

Modern Interfaces

# API

- Enterprises exchanging electronic documents
- Formalize documents -> XML
- Web Services
- Exposing core business functions

# Large Systems

- Integration
- Client/Server

# Good Software Design

- low coupling
- high cohesion

# OOD

- encapsulation
- interfaces

# Database design

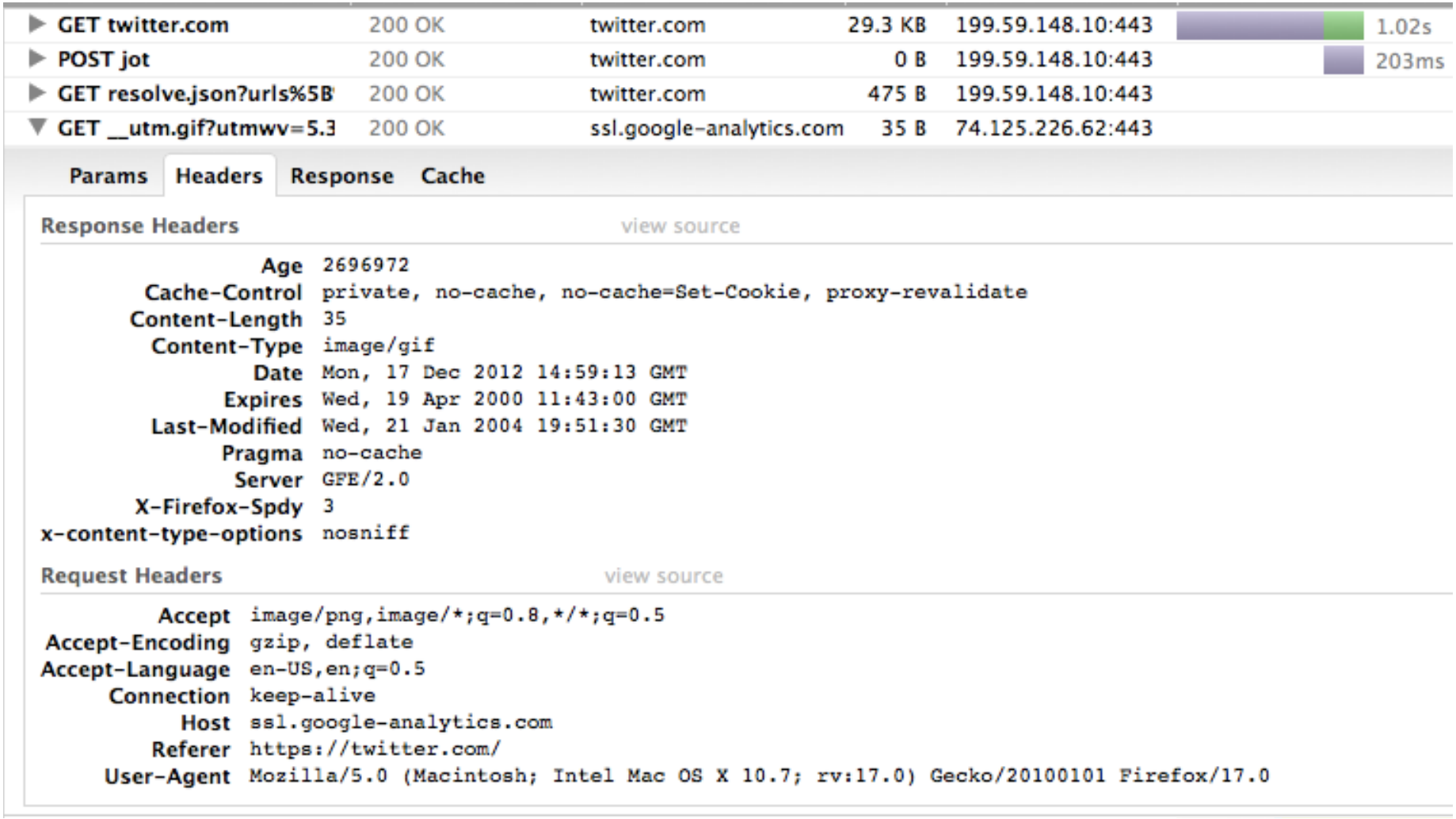
- functional dependency
- normalization

# Good API design

- Define a good Domain model
- Abstract core functions
- From DB:  CRUD
- From OO:  interfaces

# HTTP

- Transfer objects between systems
- Reliable
- bi-directional
- in-order byte stream


GET twitter.com 200 OK twitter.com 29.3 KB 199.59.148.10:443 1.02s
POST jot 200 OK twitter.com 0 B 199.59.148.10:443 203ms
GET resolve.json?urls%5B 200 OK twitter.com 475 B 199.59.148.10:443
GET __utm.gif?utmwv=5.3 200 OK ssl.google-analytics.com 35 B 74.125.226.62:443
Params Headers Response Cache
Response Headers view source
Age 2696972
Cache-Control private, no-cache, no-cache=Set-Cookie, proxy-revalidate
Content-Length 35
Content-Type image/gif
Date Mon, 17 Dec 2012 14:59:13 GMT
Expires Wed, 19 Apr 2000 11:43:00 GMT
Last-Modified Wed, 21 Jan 2004 19:51:30 GMT
Pragma no-cache
Server GFE/2.0
X-Firefox-Spdy 3
x-content-type-options nosniff
Request Headers view source
Accept image/png,image/*;q=0.8,*/*;q=0.5
Accept-Encoding gzip, deflate
Accept-Language en-US,en;q=0.5
Connection keep-alive
Host ssl.google-analytics.com
Referer https://twitter.com/
User-Agent Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:17.0) Gecko/20100101 Firefox/17.0

# Inspect

# REST

| | |
|---|---|
| C | post |
| R | get |
| U | put |
| D | delete |

# Best Practices

- think about “resources”
- elements & collections
- map out the 4 methods for each
- Prefer Nouns, Plural, Concrete
- Use Parameters for more advanced queries