

# Storing Data and Files

Thierry Sans

# Storing Data in a Database

# Modern Web Platform

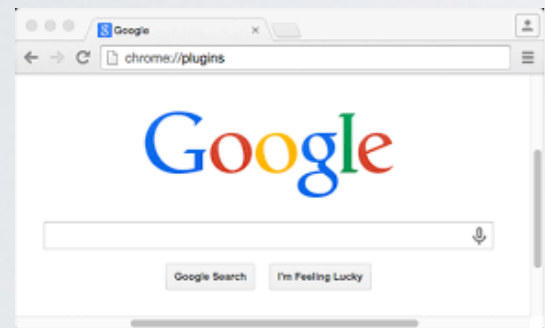
Client Side

Server Side

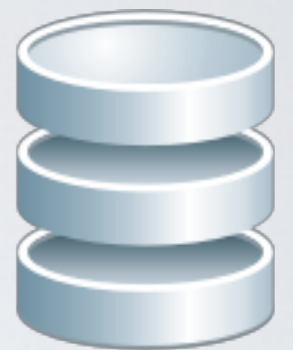


```
Macintosh HD -- top -- 80x24
Processes: 218 total, 2 running, 9 stuck, 199 sleeping, 901 threads 23:30:03
Load Avg: 1.40, 1.75, 1.00  CPU usage: 4.15% user, 4.40% sys, 91.44% idle
SharedLibs: 164K resident, 0K data, 0K incore:
MemRegions: 31278 total, 1892M resident, 117M private, 564M shared.
PhysMem: 589M used (1191M wired), 18K unused.
VM: 523K active, 1025K frameworks vsize, 0(K) swaptins, 0(K) swaptouts.
Networks: packets: 12185/8925K in, 11907/1964K out.
Disks: 88156/7285M read, 21235/425M written.

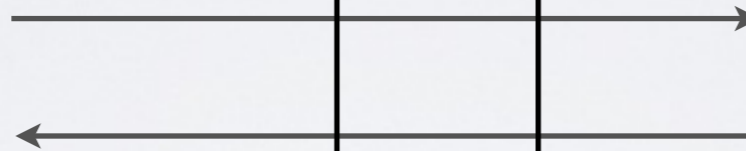
PID COMMAND %CPU TIME #TH #WQ #PORT MEM PURG CMPR PGRP PPID
592 screencapture 0.0 00:00:02.7 7 35+ 1852K+ 204K 00 212 263
598 mdworker 0.0 00:00:01.3 0 44 2032K 00 00 590 1
589 mdworker 0.0 00:00:01.3 0 44 1572K 00 00 589 1
588 top 1.7 00:00:51.1/1 0 22+ 2060K 00 00 588 584
584 bash 0.0 00:00:00.1 0 15 588K 00 00 584 583
583 login 0.0 00:00:01.3 1 20 1220K 00 00 583 482
574 auditd 0.0 00:00:00.2 0 25 560K 00 00 574 1
567 System Prefe 0.0 00:03:23.3 0 270 39M 8364K 00 567 1
561 systemstatus 0.0 00:00:03.2 1 19 1648K 00 00 561 1
559 com.apple.We 0.0 00:01:42.9 0 239 25M 00 00 559 1
558 com.apple.We 0.0 00:05:07.15 3 224 151M 1716K 00 558 1
555 bash 0.0 00:00:00.1 0 15 604K 00 00 555 554
554 login 0.0 00:00:01.3 1 20 1176K 00 00 554 481
550 bash 0.0 00:00:00.1 0 15 688K 00 00 550 549
```



Web API



Database



# Why using a database

- Persistency
- Concurrency (avoid race conditions)
- Query
- Scalability

# SQL vs NoSQL databases



# Relational database (SQL database)

Data structure	tables and tuples
Query language	SQL
Inconvenient	not-optimized for big data analysis
Advantage	complex queries
Technology	<i>PostgreSQL, MySQL, MariaDB, SQLite, MSSQL</i>

# NoSQL database

Data structure	key/value pairs
Query language	API style
Inconvenient	not adequate for complex queries
Advantage	optimized for big data analysis
Technology	<i>MongoDB, Redis, CouchDB, NeDB</i>

# ORM - Object Relational Mapping

➡ Mapping between (OOP) objects and the database structure

## Examples

- *Sequelize for PostgreSQL, MySQL, MariaDB, SQLite*
- *Mongoose for MongoDB*



# Do/Don't

- Do **retrieve selected elements only** rather than retrieving an entire collection and filtering afterwards
- Do **define primary keys** rather than relying on auto-generated ones
- Do **split data into different collections** rather than storing list attributes
- Do **create join collections** whenever appropriate (only for NoSQL database without performant join feature)

# Retrieving collections with paginated results

➔ Only retrieve what you need from a potentially large collection

## Examples

```
GET /messages [ ?page=0 ]
```

```
GET /messages?page=1
```

```
GET /messages [ ?max=100 ]
```

```
GET /messages?max=20
```

# Handling files

# Browser restrictions

- It is **impossible** to write a piece of code that reads an arbitrary file in (client-side) Javascript
- ➔ Only files selected by users through file input forms can be processed

```
<form . . . >  
    <input type="file" name="img" multiple>  
    <input type="submit">  
</form>
```

[optional] select  
multiple files

# Sending a file from the terminal

```
$ curl -X POST  
  -H "Content-Type: multipart/form-data"  
  -F "picture=@localpath/to/img.png"  
  -F "username=bart"  
  http://...
```



# Sending a file from the browser

- **Form action** (with page refresh)

```
<form action="/url"  
      method="POST"  
      enctype="multipart/form-data">
```

- **Ajax request** (without page refresh)

```
const file = document.get ...  
const formData = new FormData();  
formData.append("picture", file);  
xhr.send(formData);
```

# What is received on the server

## **File metadata**

- filename
- mimetype (file type)
- size
- and others

## **File content**

- Compressed binary or string

# MIME types

MIME (Multipurpose Internet Mail Extensions)  
is also known as the **content type**

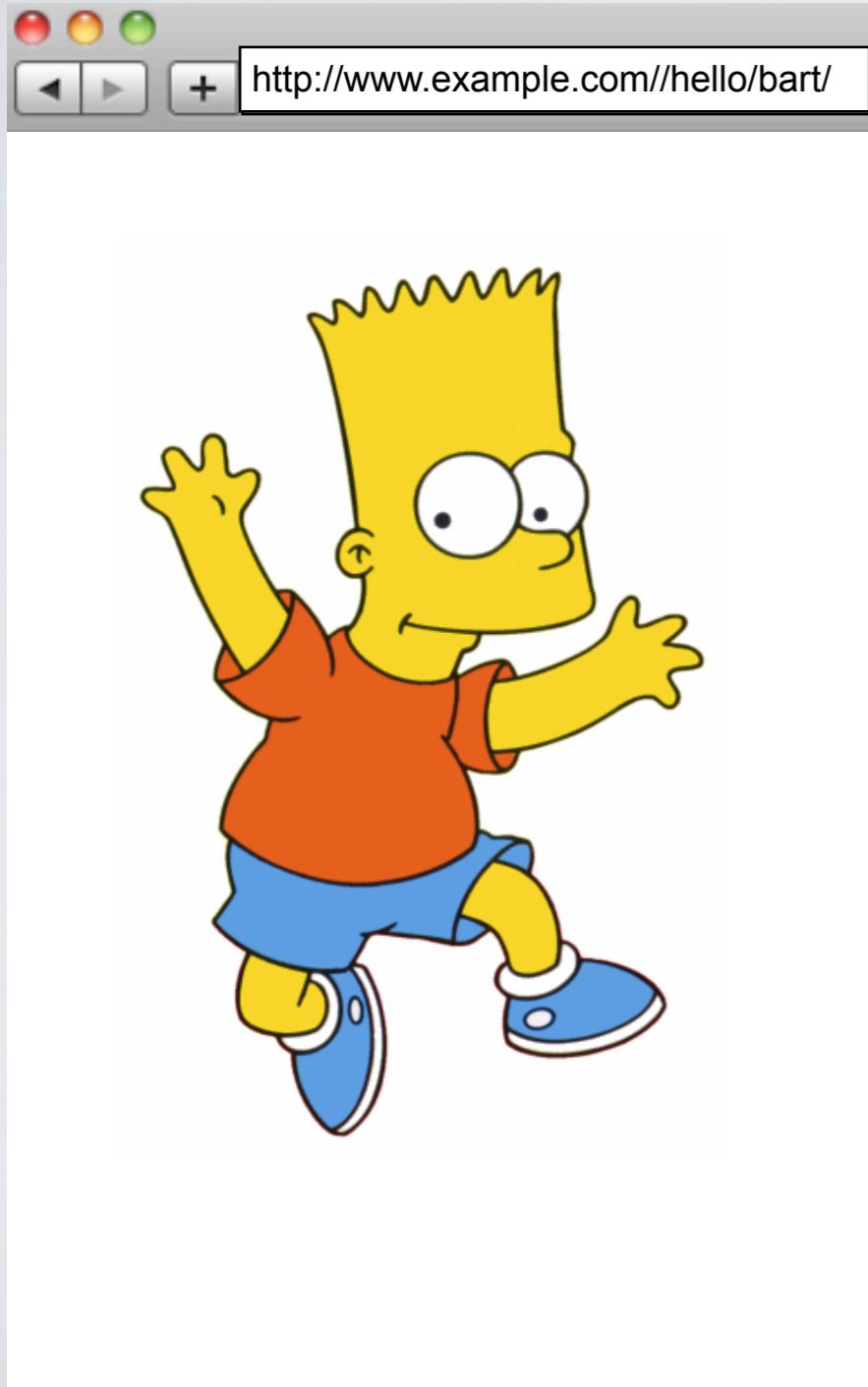
- ➔ Define the format of a document exchanged on internet (IETF standard) <http://www.iana.org/assignments/media-types/index.html>

# Examples of MIME types

- text/html
- text/css
- text/javascript
- image/jpeg - image/gif - image/svg - image/png (and so on)
- application/pdf
- application/json



# Example of how images are retrieved



*GET hello/bart/*



```
<html>
  <body>
    <img src=images/bart.jpg/>
  </body>
</html>
```

MIME : text/html



*GET images/bart.jpg*



MIME : image/jpg



# Do/Don't with files

- Do **not** send a base64 encoded file content with JSON, use `multipart/form-data` instead (compression)
- Do **not** store uploaded files with the static content
- Do **not** serve uploaded files statically (security)
- Do store the mimetype and set the HTTP response header mimetype when files are sent back