Linux and more. Data processing

Artem Trunov for Ozon Masters

На прошлом занятии

- Обрабатывали текстовые файлы
- Обрабатывали файлы в формате csv

На этом занятии

- Другие форматы данных
  - JSON
- Работа с REST API из командной строки
- Параллельная обработка

CSV vs. JSON

- Csv хорош для табличной структуры данных
- Но жизнь богаче и в ней есть место неструктурированным данным, а так же есть необходимость различать типы данных
- REST, Kafka, базы данных

- JSON
  - Стандартный формат данных, есть парсеры для множества языков
  - Совместим с Javasript (откуда произошел)
  - Типы данных - строка, число, логический, список, объект (словарь), null
  - Схема (пока черновик стандарта)

Json record


```
{
"asin": "0000031852",
"title": "Girls Ballet Tutu Zebra Hot Pink",
"price": 3.17,
"imUrl": "http://ecx.images-amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
"related": {
  "also_bought": ["B00JHONN1S", "B002BZX8Z6"],
  "also_viewed": ["B002BZX8Z6", "B00JHONN1S", "B00BFXLZ8M"],
  "bought_together": ["B002BZX8Z6"]
},
"salesRank": {"Toys & Games": 211836},
"brand": "Coxlures",
"categories": [["Sports & Outdoors", "Other Sports", "Dance"]]
}
```

О процессорах JSON

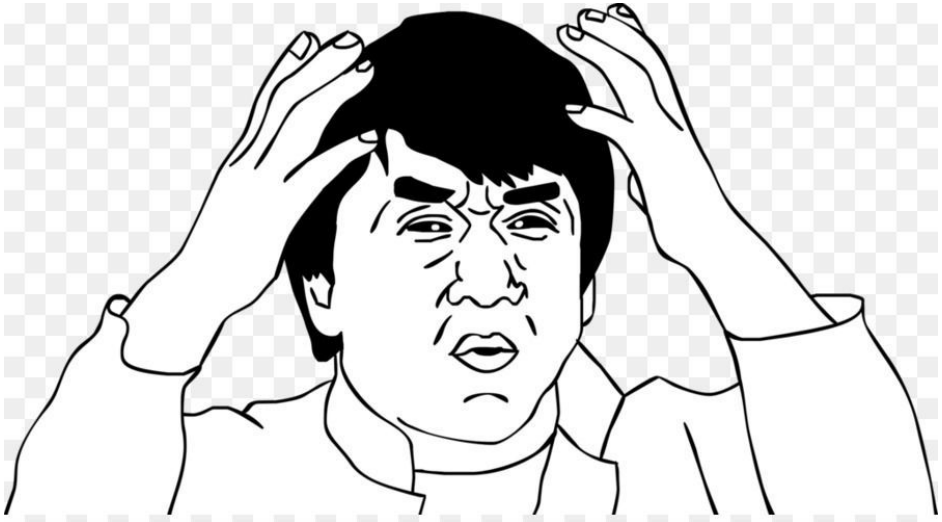
- SQL не подходит
- В XML сразу был XPath, XQuery
  - А XML был “next big thing” после Интернета.
- Для json есть несколько парсеров, каждый со своим синтаксисом.
- Почему не портировали Xpath? - неизвестно
  - Хотя, кажется, в последних спецификациях XPath, XQuery добавили поддержку JSON
  - Но реализаций нет, и это характерно.

jq

“jq is like sed for JSON data - you can use it to slice and filter and map and transform structured data with the same ease that sed, awk, grep and friends let you play with text.”

jq

“jq is like sed for JSON data - you can use it to slice and filter and map and transform structured data **with the same ease** that sed, awk, grep and friends let you play with text.”



Jq - tutorial

https://github.com/datamove/practice-repo/blob/master/tutorials/jq.md

jshon

```
$ head -1 nbagames.json | jshon -e teams -a -e players -a -e player
"Jeff Ruland"
"Cliff Robinson"
"Gus Williams"
"Jeff Malone"
...
```

JSON in Web applications

- JSON очень часто используется как формат сообщений при взаимодействии клиента с сервером (по протоколу HTTP)
  - Удобен для человека и машины
- Сами приложения, как правило, разрабатываются в парадигме REST и называются RESTful
- REST - некий набор принципов, ограничений и свойств для приложения, чтоб всем легче было работать.
  - Ресурсы и их идентификаторы
    - http://example.com/tickets
    - http://example.com/tickets/1
  - Приложение не должно иметь состояние (способствует надежности)

REST meets HTTP

- POST
  - Создает ресурс, которому назначается идентификатор
  - http://example.com/tickets
- GET
  - Запрашивает ресурс
  - http://example.com/tickets
  - http://example.com/tickets/1
- PUT
  - Изменяет ресурс целиком
- PATCH
  - Изменяет ресурс частично
- DELETE
  - Удаляет ресурс
- ***CRUD (Create-Retrieve-Update-Delete)***

HTTP

request

```
GET / HTTP/1.1
Host: www.example.com
```

header

response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 155
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close
```

header

```
<html>
 <head>
  <title>An Example Page</title>
 </head>
 <body>
  <p>Hello World, this is a very simple HTML document.</p>
 </body>
</html>
```

body

HTTP

request

```
GET / HTTP/1.1
Host: www.example.com
```

protocol version

response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 155
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
 <head>
  <title>An Example Page</title>
 </head>
 <body>
  <p>Hello World, this is a very simple HTML document.</p>
 </body>
</html>
```

response status code

HTTP мини-практика

```
datamove@linux1:~$ telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.1

HTTP/1.1 200 OK
Server: gunicorn/20.0.4
Date: Fri, 06 Nov 2020 09:33:08 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 149

        <form action="/check_login" method="post">
 <input type="text" name="login"></input>
 <input type="submit" value="Check"></input>
</form>
        Connection closed by foreign host.
```

HTTP мини-практика

```
datamove@linux1:~$ telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
POST /check_login HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

login=datamove&submit=

HTTP/1.1 200 OK
Server: gunicorn/20.0.4
Date: Fri, 06 Nov 2020 09:41:52 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 72

datamove pts/4        84.23.42.150  Fri Oct  2 09:30 - 10:03  (00:32)
Connection closed by foreign host.
```

HTTP мини-практика

```
datamove@linux1:~$ telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
POST /check_login HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

login=datamove&submit=

HTTP/1.1 200 OK
Server: gunicorn/20.0.4
Date: Fri, 06 Nov 2020 09:41:52 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 72

datamove pts/4        84.23.42.150  Fri Oct  2 09:30 - 10:03  (00:32)
Connection closed by foreign host.
```

HTTP мини-практика

```
datamove@linux1:~$ telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
POST /check_login HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

login=datamove&submit=

HTTP/1.1 200 OK
Server: gunicorn/20.0.4
Date: Fri, 06 Nov 2020 09:41:52 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 72

datamove pts/4        84.23.42.150  Fri Oct  2 09:30 - 10:03  (00:32)
Connection closed by foreign host.
```

HTTP мини-практика

```
curl -X POST http://localhost:5000/check_login -d 'login=datamove'

curl -F login=datamove http://localhost:5000/check_login

curl -X GET http://localhost:5000/check_login?login=datamove

-d - данные в запросе (payload)
-F - form data
-i - также вывести заголовки
-I - только заголовки
-H - добавить строчку к заголовку запроса
-L - follow redirection
-o - записать принятые данные в файл
-s - не показывать прогресс и ошибки
-s -S - всё таки показывать ошибки
```

HTTP response status codes

- *1xx informational response* – запрос получен, идет обработка
- *2xx successful* – запрос получен, принят, обработан
- *3xx redirection* – нужны дополнительные действия
- *4xx client error* – неправильный запрос, в т.ч. несуществующий ресурс (404)
- *5xx server error* – ошибка сервера

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

HTTP Authentication

- Простая (simple) - по имени пользователя и паролю
  - curl http://user:password@example.com
- Другие механизмы через расширения

Проблема?

- “All or nothing” (“Пан или пропал”)
  - Доступ ко всему ресурсу, всем привилегиям.
  - Нет механизма **авторизации** на отдельные части системы или отдельные действия

OAuth access token

- OAuth (2.0) - стандарт(?) для делегирования авторизации на ресурсы провайдера АПИ
  - Провайдер аутентифицирует пользователя и, с его разрешения, выдает приложению токен для доступа к определенным ресурсам пользователя, хранящимся у провайдера
  - Токен используется в заголовке http-запросов к провайдеру.
- Токен - некий аналог deploy key в Github.com
  - Он создается аутентицированным пользователем
  - Он может быть использован кем угодно
    - И, естественно, получатель токена должен беречь его, как пароль
  - Он дает ограниченные права на определенные ресурсы

Практика

- **https://github.com/datamove/practice-repo/blob/master/tutorials/REST.md**
- Работаем с Github API с помощью утилиты **curl**
- Используем **jq** для парсинга **json из** ответов API

- Получаем список issue в datamove/practice-repo
- Создаем новый тикет
- Изменяем тикет
- Пытаемся его удалить

Обсуждение домашки

Параллельное исполнение

Find -exec

- Надо найти какие-то файлы и что-то сделать с ними
- Find /etc/ssh/ -name ssh_host* -exec ls -al {} \;
- Find /etc/ssh/ -name ssh_host* -exec cp {} {}.back \;

xargs

- Получает список со стандартного ввода и передает его в качестве аргумента приведенной команды.
- `find /etc/ssh/ -name ssh_host*pub | xargx ls -al`
  - Тоже что `ls -al  /etc/ssh/ssh_host*pub`
  - Только учитывает предел по числу аргументов!
- `find /etc/ssh/ -name ssh_host*pub | xargx -n 2 ls -al`
- `find /etc/ssh/ -name ssh_host*pub | xargx -I '{}' cp  {} /tmp`

-n - сколько аргументов передавать
-P - сколько потоков
-I '{}' - заменять данный шаблон на аргумент

GNU parallel

- GNU parallel - параллельное исполнение, в том числе на разных хостах
- Xargs на стероидах
- Очень много опций (ещё одна вселенная)

```
ls *.wav | parallel lame {} -o {}.mp3
cat urllist | parallel -P+2 'wget "{}" -O - | python3 parse.py'
```

```
man parallel
```

#3700 lines(!)
http://www.youtube.com/playlist?list=PL284C9FF2488BC6D1
https://www.gnu.org/software/parallel/parallel_tutorial.html
https://www.gnu.org/software/parallel/parallel_design.html