ripple motion
services

Créateur d’applications mobiles

TECH TALK

GRATUITS & OUVERTS AU PUBLIC

Tech talk sous forme de courtes formations
le **mercredi** de **17h** à **18h** !

N’hésitez pas à nous **contacter** afin de participer
aux **prochaines sessions** au sein de nos locaux !

ripple motion services

TECH TALK

ANIMÉ PAR

GHISLAIN DEFFRASNES

DÉVELOPPEUR

ghislain.deffrasnes@ripplemotion.fr

ripple motion services

TECH TALK #16 – 22.03.2017

TIMEZONES

WITH PYTHON

ripple motion services

GHISLAIN DEFFRASNES
DÉVELOPPEUR

UTC VS GMT

almost the same thing

- GMT **Greenwich Mean Time** is the ancestor of UTC
- UTC **Coordinated Universal Time** adopted by scientific community in 1972
- norme ISO 8601 created in 1988. Ex :
  - 1977-04-22T01:00:00-05:00
  - 1977-04-22T06:00:00Z

GHISLAIN DEFFRASNES
DÉVELOPPEUR

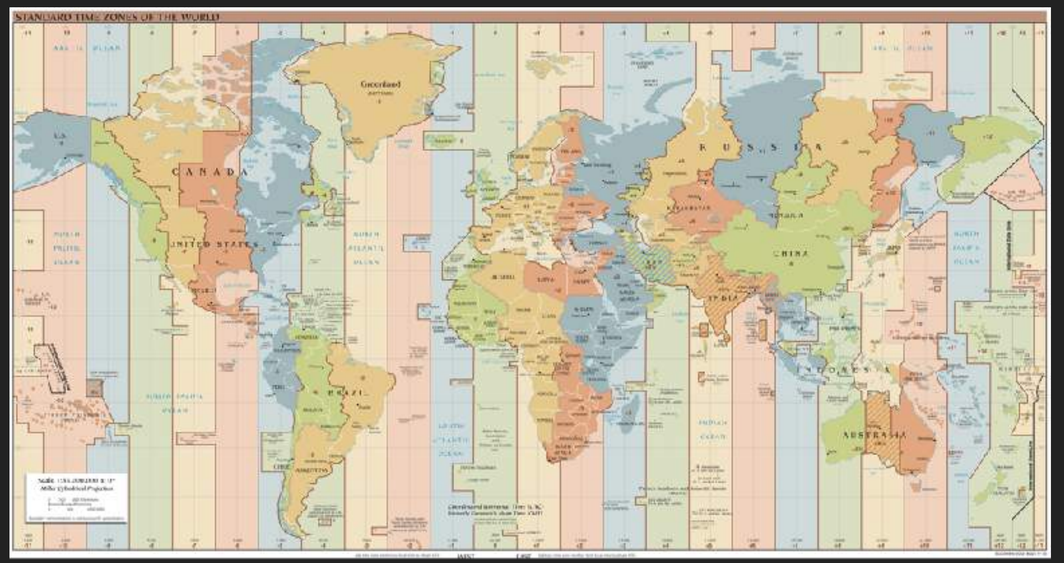
WHAT IS A TIMEZONE?

Examples

- Paris UTC+01:00 CET (Central European time zone)
- Beijing UTC+08:00 (China time zone)
- San Francisco UTC-08:00 PT/PDT (Pacific time zone)

+ 7H / PARIS

- 9H / PARIS



GHISLAIN DEFFRASNES
DÉVELOPPEUR

DST?

Daylight Saving Time

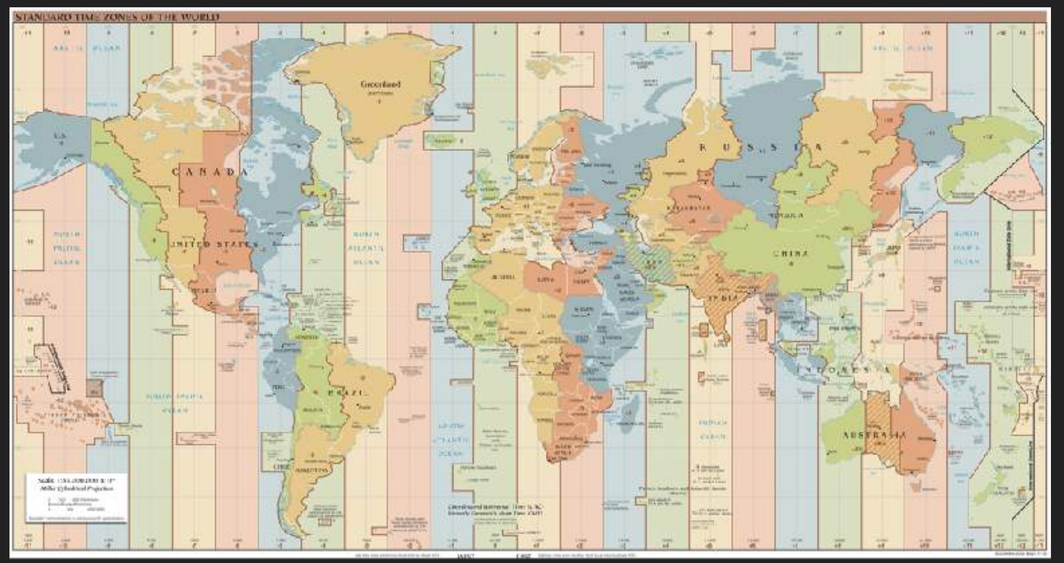
- Paris UTC+02:00
- Beijing UTC+08:00 : No DST — + 6H / PARIS
- San Francisco UTC-07:00 — - 9H / PARIS

Change dates :

- 26 March 2017

⚠

- 13 March 2017



AMBIGUOUS AND NON-EXISTENT DATE TIMES

GHISLAIN DEFFRASNES
DÉVELOPPEUR

- Existent : 2017-03-26 02:15 in UTC timezone
- Non-existent : 2017-03-26 02:15 in Paris timezone ✅
- Non-ambiguous : 2017-10-29 02:15 in UTC timezone ❌
- Ambiguous : 2017-10-29 02:15 in Paris timezone ✅ ❌

More complex : countries may have changed also their DST rule in the passed !

GHISLAIN DEFFRASNES
DÉVELOPPEUR

TIME & DATETIME MODULES

Modules

- time :
  - to manipulate OS-defined times
  - to manipulate timestamps
- datetime : to manipulate date, time, datetime and time delta objects

```
#!/usr/bin/env python

import datetime

import time
```

GHISLAIN DEFFRASNES
DÉVELOPPEUR

TIME MODULE

```
#!/usr/bin/env python

import time
```

- time.time() gives current time's UTC timestamp
- time.gmtime(0) gives the Epoch, "first point of time" (time zero)
  - => 1st of January 1970 on UNIX systems
- time is updated 50-100 times a second

=> Don't use time module !

Because its depend on your machine settings.

GHISLAIN DEFFRASNES
DÉVELOPPEUR

DATETIME MODULE

```
#!/usr/bin/env python

import datetime
```

- Gives types of objects :
  - datetime.date(year, month, day)
  - datetime.time(hour, minute, second, microsecond, **tzinfo**)
  - datetime.datetime(...) [see datetime.combine(date, time)]
  - datetime.timedelta (duration)
- => All immutable
- datetime.tzinfo class
  - Defines how the conversion UTC to a timezone should be computed
  - and the reverse

```
>>> from datetime import time, tzinfo, timedelta
>>> class GMT1(tzinfo):
...     def utcoffset(self, dt):
...         return timedelta(hours=1)
...     def dst(self, dt):
...         return timedelta(0)
...     def tzname(self,dt):
...         return "Europe/Prague"
```

Return offset of local time from UTC

Return DST adjustment

Return timezone name

GHISLAIN DEFFRASNES
DÉVELOPPEUR

TIMEZONE RELATED OBJECTS

time or datetime may be naive or aware

- aware : dt.tzinfo is not None and dt.tzinfo.utcoffset(d) does not return None
- naive : not aware

Comparing naive with aware datetimes results in TypeError !

=> Avoid using naive datetimes and times

GHISLAIN DEFFRASNES
DÉVELOPPEUR

PYTZ

World Timezone Definitions for Python

pip install pytz

- Good practice :
  - always work in UTC
  - convert to localtime when generating output to be read by humans
  - please avoid using tz.normalize() method
- Do not instantiate datetime with tzinfo (except with pytz.utc)
- Use pytz.utc and pytz.timezone('Continent/City') to instantiate a tz
- Use tz.localize(dt) to add a timezone on naive datetime already expressed in this tz
- Use dt.astimezone(tz2) to convert aware datetime from a timezone to another timezone

PYTZ PRACTICE

GHISLAIN DEFFRASNES
DÉVELOPPEUR

```python
#!/usr/bin/env python

import datetime
import pytz

pytz.common_timezones

pytz.country_names
pytz.country_timezones


fmt = '%Y-%m-%d %H:%M:%S %Z%z'
tz1 = pytz.timezone('Europe/Paris')
    => <DstTzInfo 'Europe/Paris' LMT+0:09:00 STD>
naive_dt = datetime.datetime(2017, 3, 22, 17, 15, 0)

# Naive value from tz1 to aware value tz1
aware_dt = tz1.localize(naive_dt)
    => datetime.datetime(2017, 3, 22, 17, 15, tzinfo=<DstTzInfo 'Europe/Paris' CET+1:00:00 STD>)
aware_dt.strftime(fmt)
    => '2017-03-22 17:15:00 CET+0100'

# Set the tzinfo class to datetime does not work
datetime.datetime(2017, 3, 22, 17, 15, 0, tzinfo=tz1)  # or naive_dt.replace(tzinfo=tz1)
    => datetime.datetime(2017, 3, 22, 17, 15, tzinfo=<DstTzInfo 'Europe/Paris' LMT+0:09:00 STD>)

# Aware UTC datetime
datetime.datetime(2017, 3, 22, 17, 15, 0, tzinfo=pytz.utc)  # ok for utc
    => datetime.datetime(2017, 3, 22, 17, 15, tzinfo=<UTC>)
pytz.utc.localize(datetime.datetime(2017, 3, 22, 17, 15, 0))
    => datetime.datetime(2017, 3, 22, 17, 15, tzinfo=<UTC>)
```

```python
# Aware value from tz1 to aware value tz2
tz2 = pytz.timezone('Asia/Shanghai')
aware_dt2 = aware_dt.astimezone(tz2)
    => datetime.datetime(2017, 3, 23, 0, 15, tzinfo=<DstTzInfo 'Asia/Shanghai' CST+8:00:00 STD>)
aware_dt2.strftime(fmt)
    => '2017-03-23 00:15:00 CST+0800'

# astimezone() on naive datetime is forbidden
naive_dt.astimezone(tz1)
    => ValueError raised !


# Ambiguous
dt_ambiguous = datetime.datetime(2017, 10, 29, 2, 15)
tz1.localize(dt_ambiguous)
    => datetime.datetime(2017, 10, 29, 2, 15, tzinfo=<DstTzInfo 'Europe/Paris' CET+1:00:00 STD>)
# No error ! The system has chosen for you.
tz1.localize(dt_ambiguous, is_dst=None)
    => pytz.AmbiguousTimeError raised !

# Non existent
dt_non_existent = datetime.datetime(2017, 3, 26, 2, 15)
tz1.localize(dt_non_existent)
    => datetime.datetime(2017, 3, 26, 2, 15, tzinfo=<DstTzInfo 'Europe/Paris' CET+1:00:00 STD>)
tz1.localize(dt_non_existent, is_dst=None)
    => pytz.NonExistentTimeError raised !

# pytz.NonExistentTimeError and pytz.AmbiguousTimeError inherit from pytz.InvalidTimeError
```

ripple motion services

GHISLAIN DEFFRASNES
DÉVELOPPEUR

TIMESTAMPS

UNIX timestamp

- timestamp => datetime : use

  datetime.datetime.fromtimestamp(timestamp, tz) with a tz != None

  - gives an aware datetime
- datetime => timestamp : use
  - Python <3.3 (aware_dt - datetime.datetime(1970, 1, 1, tzinfo=pytz.utc)).total_seconds()
  - Python 3.3+ aware_dt.timestamp()

DJANGO UTILS TIMEZONE

from django.utils import timezone

GHISLAIN DEFFRASNES
DÉVELOPPEUR

- timezone.get_default_timezone() gives tzinfo instance for timezone defined in settings (when USE_TZ == True)
- timezone.get_current_timezone() gives current timezone
  - use timezone.activate() and deactivate() to set the current timezone
- timezone.now() gives UTC-converted aware datetime from default timezone (when USE_TZ==True)
- Naive / aware helpers
  - timezone.is_aware()
  - timezone.is_naive()
  - timezone.make_naive()
  - timezone.make_aware()
  - timezone.localtime()

GHISLAIN DEFFRASNES
DÉVELOPPEUR

REFERENCES

- https://www.timeanddate.com
- https://docs.python.org/2/library/time.html
- https://docs.python.org/2/library/datetime.html
- http://pytz.sourceforge.net/
- https://docs.djangoproject.com/en/1.10/ref/utils/#module-django.utils.timezone
- http://dateutil.readthedocs.io/en/stable/parser.html

MERCI DE VOTRE PARTICIPATION !

ripple motion services

ripple motion
services

Créateur d’applications mobiles