雰囲気実装を少し抜け出そう！ RFCからPHPの実装までを考えるタイムゾーンとサマータイム！！！ のおまけ資料

登壇資料の走り書きです。みても面白くないかもしれません。

自己紹介

所属

株式会社TechBowl

住んでるところ

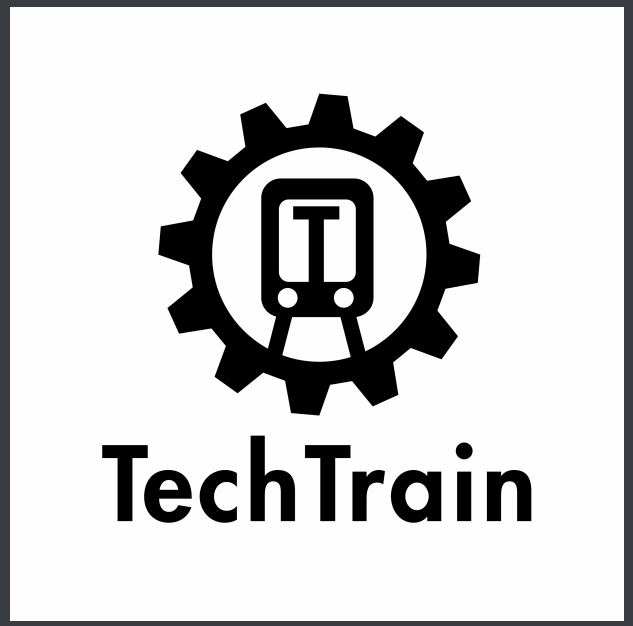
東京

何やってる?

「TechTrain」というサービスで反復横跳びし続けている何でも屋さん(Laravel, Next.js, AWS, etc...)

趣味

- お酒(よく溺れる)
- サウナ
- 読書



TechTrain

エンジニア教育+Directスカウトのサービス。
Coding Stoicをテーマに「うるせえコードかけ！」と言いがちなメンターが多めのエンジニアを育てるためのサービスです。

TechTrain

一緒に働いてくれる人を探しています！

1. バックエンドエンジニア(Laravel + DDD)

2. フロントエンドエンジニア(Next.js with TypeScript)

3. TechTrainのメンター -> 筋がいい人なら教えたいぜ！

ISO 8601とRFC3339は特定の時刻と時間を表現するための標準規格です。例：2020-09-06T17:35:24.485Z。ここで Z は UTC を表す接尾辞です。

https://ja.wikipedia.org/wiki/ISO_8601

https://datatracker.ietf.org/doc/html/rfc3339

そもそもtimestampって何?

以前は、夏時間の期間に入るまたは終わる度に手動でコンピュータに内蔵されている時計の時刻を合わせていたが、近年のオペレーティングシステムは、自動的に内蔵時計を修正する機能をもっている。ファイルのタイムスタンプの扱いは、使用するファイルシステムおよびオペレーティングシステムによって異なる。

例えば、FATのようなタイムスタンプの記録にローカルタイムを利用するファイルシステムの場合、夏時間内で修正されたファイルを、夏時間外で読み込んだ場合、時刻が1時間ずれる。一方、NTFSのような、タイムスタンプを協定世界時（UTC）で記録するファイルシステムを利用している場合、このような問題は起きない。

時刻の内部管理にUTCを使うことにより、夏時間を意識せずにファイルの読み書きができるものの、オペレーティングシステム上での取り扱いは、各システムによって異なる。

Laravelだとすればdefaultはtimestamp型のカラムになっているので、そうしますが、実は問題があります。
そう2038年問題です。

2038年以降については動作が保証されないため、datetime型で雰囲気UTCを前提で使うことになります。

Windows系の場合

FATのようなタイムスタンプの記録にローカルタイムを利用するファイルシステムであるため、
Windows XP以前のOSでは、時刻は現在有効な標準時に合わせて表示される。
例えば、夏時間の期間中にタイムスタンプが9時であった場合、期間外では10時と表記される。
この方式では表示される時刻が実際の時刻と異なることがある。一方で、夏時間の期間の前後で時刻に不連続が発生しなくなるという利点がある。また、時代・地域による期間・調整時間の差異や、未来の時刻を取り扱う時に実施当日までに変更される可能性がある夏時間規則を考慮する必要がない。

WindowsでもWindows 7以降およびmacOSの場合

NTFSのような、タイムスタンプを協定世界時（UTC）で記録するファイルシステムを利用しているため、
期間中に9時であったものは、期間外になっても9時と表記される。この方式の利点・欠点は前者の逆である[24]

LinuxやBSD系オペレーティングシステムではtz databaseを用いて夏時間を管理している。

デジタルカメラなどの画像ファイルで使われるEXIFではGPS関連の項目を除いてUTCやタイムゾーンなどは考慮されていない。このため、夏時間を採用している地域では、画像を読み込む時期によって撮影時刻の記録・表示が1時間ずれる。

向への変換に対応する関数と考えられます。
（なぜ exact -> local の変換は 1 対 1 なのに、local -> exact への変換は曖昧なのかの理由は後述します）。

IANA Time Zone Database（TZ database とも呼ばれる）を使用します。
これは、タイムゾーン関数の世界的なリポジトリであると考えられます。各 IANA タイムゾーンは以下のものを持ちます:

time zone ID は、地理的な範囲を市などで表したものです（例）。例えば Europe/Paris、Africa/Kampala などです。また、1 つのタイムゾーンオフセットも表すこともできます。例えば UTC（+00:00 オフセットの定数）や Etc/GMT+5（歴史的な経緯でこれは-05:00の意味）
タイムゾーンオフセット定義 は、すべての UTC のオフセットのデータベースで、1970 年 1 月 1 日からのすべての情報が記載されています（例）。これは、ある UTC の範囲（未来の範囲も含まれる）を特定のオフセットに写像するテーブルであると考えられます。いくつかのタイムゾーンは、オフセットが一時的に変わったりすることがあります。例えば サマータイム が導入されていると、春の初めと秋の終わりでオフセットが年に 2 回も変更されます。オフセットは、国がタイムゾーンを変更するよう

Time Zone Databaseにより定義された各タイムゾーン

場合によってはUTCからの時間の差分（協定世界時との差）を複数持つ
典型的には、標準時と夏時間双方が同一のタイムゾーンに含まれる。

考慮しなくてはいけないパターン

1. サマータイムでずれた時のインサートなどの日時の取り扱い
2. タイムゾーン自体が政治的な問題でずれてしまった時の取り扱い->この辺り書いてもしょーがないので、やめる

サマータイムに入る時、何が起こっているのか。

見た目: 時計が 1 時間進みます
実際: 時刻ではなくオフセットが動いている

オフセットが+1されて、サマータイムが終わる時には、オフセットが-1される。

仮に日本でサマータイムに入る時

```
- 2024-03-07T00:00:00Z+09:00
+ 2024-03-07T00:00:00Z+10:00
```

仮に日本でサマータイムが終わる時

```
- 2024-03-07T00:00:00Z+10:00
+ 2024-03-07T00:00:00Z+09:00
```

新たな Date/Time Format が来るかもしれない

Internet Extended Date/Time Format (IXDTF)

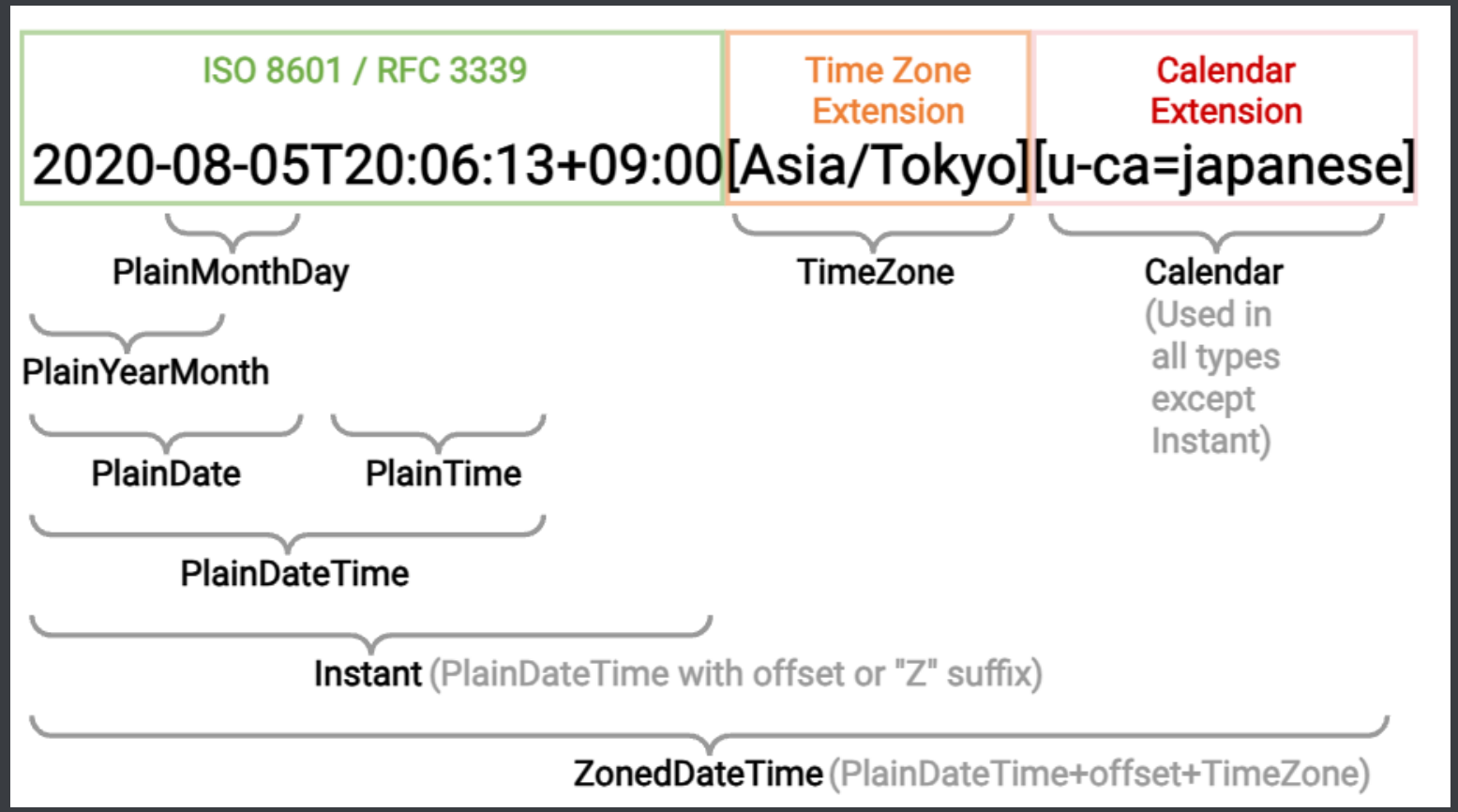
RFC Draft: Date and Time on the Internet: Timestamps with additional information にて新たに Internet Extended Date/Time Format (IXDTF) が提案されている。

要点

1. RFC 3339 の Internet Date/Time Format にタイムゾーンと、キーバリューペアを持つタグを付加できるようになる
2. 2024/03/07現在、まだDraftとなっている
3. 2のこともあり、まだPHPもCarbonも未対応

※ECMAScriptのTemporalはすでに仕様の議論に入っている模様。

全体構造


ISO 8601 / RFC 3339
Time Zone Extension
Calendar Extension
2020-08-05T20:06:13+09:00[Asia/Tokyo][u-ca=japanese]
PlainMonthDay
TimeZone
Calendar
(Used in all types except Instant)
PlainYearMonth
PlainDate
PlainTime
PlainDateTime
Instant (PlainDateTime with offset or "Z" suffix)
ZonedDateTime (PlainDateTime+offset+TimeZone)

Draftである理由

```
from datetime import datetime
datetime.fromisoformat("2022-07-08T00:14:07Z[!Europe/London]")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: Invalid isoformat string: '2022-07-08T00:14:07Z[!Europe/London]'
```

Pythonの例が記述されており、このように移行のための対応が言語やライブラリでなされるまで、移行自体に問題が出る可能性があるため、ドキュメント自体に移行についての記述がなされるまでは、「Has Issues」のステータスとして取り扱われている。

うるう秒なくなったよーの話も面白いかもしれない。
Googleの謎技術の話w
1秒を薄めていくやつ

1. 各国の標準時のもとになる「協定世界時（UTC）」も、世界中にある400台以上の原子時計の進み具合を平均する
2. うるう秒は、そんなUTCと地球の自転に基づく時刻が、大きくずれないよう合わせるためのものだ。
3. 具体的には時間差が0・9秒を超えそうになったとき、UTCを1秒分調整する。精密な原子時計と地球の自転速度のムラの「つじつま合わせ」とも言える。
4. 似たような言葉に「うるう年」があるが、根本的に違う。
5. うるう年は地球が太陽の周りをまわる公転周期が365日より若干長いことから、ほぼ4年に1回、規則的にくる。一方、うるう秒は、不規則なうえ、半年前にしか分からない。

毎回、地球の自転を観測する国際機関が調整のタイミングを決める。それを受けて、日本では標準時を管理する情報通信研究機構（NICT）などが周知や通報にあたる。これまでに27回、直近だと17年にあった。

1972年に閏秒は導入されている。

POSIX とは

POSIX (Portable Operating System Interface) は、UNIX 系 OS 間でアプリケーションの移植性を高めるために定義された IEEE の標準規格です。簡単に言うと「OS の互換性のための標準規格」です。

https://zenn.dev/pixiv/articles/23b726da2236cd

https://datatracker.ietf.org/doc/draft-ietf-sedate-datetime-extended/

https://tc39.es/proposal-temporal/docs/strings.html#machine-readable-string-persistence-overview

https://qiita.com/yamato225/items/676beef96efb8751e581

https://www.casio.com/jp/watches/contents/daylight-saving/

https://zenn.dev/mpyw/articles/laravel-datetimezone-best-practices

https://dev.mysql.com/doc/refman/8.0/ja/datetime.html

https://mysql.sql55.com/sql/mysql-now-curdate.php#:~:text=MySQL で現在の日時を取得するには,文字列として返します。

https://tech.hajimari.inc/entry/2022/09/05/100000

https://dev.mysql.com/doc/refman/8.0/ja/datetime.html

https://www.cluemediator.com/how-to-identify-daylight-saving-time-in-php

https://stackoverflow.com/questions/1646171/mysql-datetime-fields-and-daylight-savings-time-how-do-i-reference-the-extra

https://stackoverflow.com/questions/45313886/laravel-carbon-not-factoring-in-daylight-savings-time

ライブラリ候補

https://github.com/brick/date-time
https://github.com/briannesbitt/Carbon
https://github.com/fre5h/datetime-php
https://github.com/aeon-php/calendar

タイムゾーンとは

タイムゾーンとは

- 地球上の異なる地域で標準時を基準にした時間帯を指す
- 地球は24時間で一周するため、世界を24の時間帯に分ける
- 標準時または協定世界時（UTC）を基準とする

そもそもなぜタイムゾーンが必要とされたのか?

そもそもなぜタイムゾーンが必要とされたのか?

1. 19世紀後半に鉄道の旅や通信技術の進歩に伴って生じた、地域ごとの時刻の違いによる混乱に対処するためだった
2. タイムゾーンが導入される以前は、各都市や町は太陽の位置を基準にした独自の現地時間を持っていた
3. そのため、異なる場所でのイベントのスケジュールや調整が困難だった

phpはどこから日付の値を取得しているのか?

そもそもですが、PHPはC言語で書かれているため、取得処理をC言語のtimeライブラリに依存しています。

どこからtimezoneの情報を取得しているのか?

- `ext/date/lib/timezonedb.h`
- `ext/date/lib/timezonemap.h`

というのがあってそこでマッピングなどを行っている。
IANA（Internet Assigned Numbers Authority）に基づいていて、定期的に更新されているらしい。
さらっとみた限りだと、一番最後のCommitは「2019-6-30 23:45」だった。

Q. 正確な日時が必要とされるシステムではどうするのか?

A. NetworkTimeProtocol (NTP)を使って、時刻同期を行う

- 上記については意識的に行いつつ、精度を確認するための仕組みを提供しているベンダーもある
    - AWSの場合は、Amazon Time Sync Service

RFCにおけるサマータイムについて調べる

1. wall-clock time（ローカル時間 や clock time とも呼ばれる、タイムゾーンに依存した時刻）
2. exact time（UTC 時間 とも呼ばれる、地球上のどこでも同じ時刻）を区別すること

Coordinated Universal Time （UTC）は、世界が時計と時刻を調整、規制するための時間の基準です。
これは、経度 0 度における平均太陽時から 1 秒以内のものであり、サマータイムによる調整は行われていません。
これは実質的にグリニッジ標準時（GMT）の後継です。
サマータイムの調整が行われているのは、wall-clock timeのみ。