Правильная работа с часовыми поясами в Rails-приложении

19 июня 2015 г.

Новиков Андрей, ведущий разработчик в

AT Consulting

DevConf

Проблемы?

Признаки того, что у вас не всё в порядке с обработкой времени:

- Пост, опубликованный в час ночи, уехал в календаре на вчера?
- Времена в приложении скачут на три часа туда и обратно?
- Пользователи яростно хотят видеть всё в своём времени и негодуют, что всё по Москве?

Часовой пояс — это...

- Смещение от UTC (например −12:00 или +13:45)
- Правила перевода стрелок на летнее время и обратно
- История изменений (когда и куда переводили стрелки)

tzdata

- Так же известна как база данных часовых поясов Олсона
- Идентификатор часового пояса — строка вида Регион/Место
  - Мы сейчас в Europe/Moscow
- Хранит всю историю о часовых поясах с 1 января 1970
- Используется в *nix системах, большинстве СУБД и языков
- Стандарт де-факто, если вы не на Windows :-)

ActiveSupport::TimeZone

- Обёртка над часовым поясом из TZ Database (гем tzinfo)
- Предоставляет методы для разбора времени в контексте данного часового пояса
- Использует свои имена для обозначения часовых поясов (Moscow вместо Europe/Moscow, но есть маппинг)
- Набор часовых поясов по умолчанию… странный

```
time_zone = ActiveSupport::TimeZone['Novosibirsk']
```

ActiveSupport::TimeZone — основные методы

- #now возвращает текущее время в данном часовом поясе
  - `time_zone.now # => Fri, 19 Jun 2015 15:50:00 NOVT +06:00`
- #parse(string) парсит время и переводит его в этот часовой пояс, умеет учитывает летнее время и смещение от UTC
  - `time_zone.parse("2015-06-19T12:50:00")`
    `# => Fri, 19 Jun 2015 12:50:00 NOVT +06:00`
- #at переводит Unix timestamp во время в часовом поясе
- #local(*args) позволяет составить время из компонентов
- И многие другие!

Прочие методы для работы с временем

- `Time.zone` возвращает объект часового пояса который сейчас используется для обработки запроса (используется глобально)
- `Time.zone_default` — часовой пояс из config/application.rb
- `Time.with_zone(&block)` меняет `Time.zone` внутри блока.
- `Time.current` и `Date.current` работают в часовом поясе приложения, а не ОС, в отличие от `Time.now` и `Date.today`
- `Time#in_time_zone(tz)` переводит время в заданный часовой пояс (принимает как объект, так и идентификаторы

```
Time.parse('2015-06-19T12:50:00').in_time_zone('Asia/Tokyo')
# => Fri, 19 Jun 2015 18:50:00 JST +09:00
```

HOWTO: Переключение всего приложения на часовой пояс текущего пользователя

```ruby
class ApplicationController < ActionController::Base

  around_action :with_time_zone, if: 'current_user.try(:time_zone)'

  protected

  def with_time_zone(&block)
    time_zone = current_user.time_zone
    logger.debug "Используется часовой пояс пользователя: #{time_zone}"
    Time.use_zone(time_zone, &block)
  end

end
```

Загрузка часового пояса из базы

```ruby
# Инициализирует объект класса +ActiveSupport::TimeZone+ для работы с
# часовым поясом, хранящимся в БД как идентификатор TZ database.
def time_zone
  unless @time_zone
    tz_id = read_attribute(:time_zone)
    as_name = ActiveSupport::TimeZone::MAPPING.select do |_,v|
      v == tz_id
    end.sort_by do |k,v|
      v.ends_with?(k) ? 0 : 1
    end.first.try(:first)
    value = as_name || tz_id
    @time_zone = value && ActiveSupport::TimeZone[value]
  end
  @time_zone
end
```

Сохранение часового пояса в базу

```
# Сохраняет в базу данных идентификатор часового пояса из TZ Database,
# у объекта устанавливает часовой пояс — объект +ActiveSupport::TimeZone+
def time_zone=(value)
  tz_id   = value.respond_to?(:tzinfo) && value.tzinfo.name || nil
  tz_id ||=
TZInfo.Timezone.get(ActiveSupport::TimeZone::MAPPING[value.to_s] ||
value.to_s).identifier rescue nil
  @time_zone = tz_id &&
ActiveSupport::TimeZone[ActiveSupport::TimeZone::MAPPING.key(tz_id) ||
tz_id]
  write_attribute(:time_zone, tz_id)
end
```

Поддержка часовых поясов в PostgreSQL

- Работает с часовыми поясами из tzdata «из коробки»:
  ```
  SELECT '2015-06-19T12:13:14Z' AT TIME ZONE 'Europe/Moscow';
  ```
- Может интерпретировать время как в UTC либо как локальное.
- Типы timestamp и прочие хранят данные без обработки.
- Те же типы, но с «with time zone» в названии не хранят часовой пояс, а только время в UTC и автоматически его конвертируют!
- Резюме: в целом неплохо, но есть подводные камни.

Поддержка часовых поясов в MySQL

- Тоже работает с tzdata, но из коробки данных может и не быть:
```
SELECT CONVERT_TZ('2015-06-19 12:13:14', 'UTC', 'Europe/Moscow');
```
- Тип datetime хранит время как есть, никак не обрабатывает.
- Тип timestamp автоматически конвертирует значение в UTC для хранения и обратно в локальное время для отображения
- Резюме: примерно так же — жить можно.

HOWTO: Выборка всех записей за дату

- Rails при подключении к СУБД устанавливает часовой пояс в UTC, и все времена хранятся тоже в UTC, поэтому запрос:
```
News.where('published_at >= ? AND published_at <= ?',
Date.today, Date.tomorrow)
```
не вернёт записи за первые три часа суток (UTC+3, все дела)
- Необходимо прямо указать момент времени в нужном часовом поясе, чтобы ActiveRecord его правильно сконвертировал:
```
News.where('published_at >= ? AND published_at <= ?',
Time.now.beginning_of_day, Time.now.end_of_day)
```

HOWTO: Добавление недостающих поясов

- config/initializers/timezones.rb

```
ActiveSupport::TimeZone::MAPPING['Simferopol']    = 'Europe/Simferopol'
ActiveSupport::TimeZone::MAPPING['Omsk']          = 'Asia/Omsk'
ActiveSupport::TimeZone::MAPPING['Novokuznetsk']  = 'Asia/Novokuznetsk'
ActiveSupport::TimeZone::MAPPING['Chita']         = 'Asia/Chita'
ActiveSupport::TimeZone::MAPPING['Khandyga']      = 'Asia/Khandyga'
ActiveSupport::TimeZone::MAPPING['Sakhalin']      = 'Asia/Sakhalin'
ActiveSupport::TimeZone::MAPPING['Ust-Nera']      = 'Asia/Ust-Nera'
ActiveSupport::TimeZone::MAPPING['Anadyr']        = 'Asia/Anadyr'
```

- config/locales/ru.yml

```
ru:
  timezones:
    Simferopol:     Республика Крым и Севастополь
    Omsk:           Омск
    Novokuznetsk:   Новокузнецк
    Chita:          Чита
    Khandyga:       Хандыга
    Sakhalin:       Сахалин
    Ust-Nera:       Усть-Нера
    Anadyr:         Анадырь
```

- https://gist.github.com/Envek/cda8a367764dc2cacbc0

HOWTO: Select из российских часовых поясов

- config/initializers/timezones.rb

```ruby
class ActiveSupport::TimeZone
  @country_zones  = ThreadSafe::Cache.new

  def self.country_zones(country_code)
    code = country_code.to_s.upcase
    @country_zones[code] ||=
      TZInfo::Country.get(code).zone_identifiers.select do |tz_id|
        MAPPING.key(tz_id)
      end.map do |tz_id|
        self[MAPPING.key(tz_id)]
      end
  end
end
```

- Где-то в app/views

```
= f.input :time_zone, priority: ActiveSupport::TimeZone.country_zones(:ru)
```

- https://github.com/rails/rails/pull/20625

А что же фронтенд?

- Никаких часовых поясов, смещение от UTC и ничего более.
- Всё остальное — только внешними библиотеками.
- Пока что лучше всех Moment Timezone (включает в себя tzdata)
  - `moment("2015-06-19T10:05:00Z").tz('Europe/Moscow')`
  - `moment.parseZone("2015-06-19T13:05:00+03:00")`
- Для больших фреймворков, смотрите библиотеки, обеспечивающие эту всю красоту, такие как angular-moment

А в Новосибирске снова ноябрь...

- Если выполнить в js `new Date();` и результат отправить в Rails:
  `Time.parse('Mon May 18 2015 22:16:38 GMT+0600 (NOVT)')`
  вернёт 2015-11-01 22:16:38 +0600
- Решение — использовать формат по стандарту ISO8601:
  `Time.iso8601('2015-05-18T22:16:38+06:00')`
  выдаст ожидаемое 2015-05-18 22:16:38 +0600
- Разрабатывая в Москве вы никогда не обнаружите этот баг!
- Установите на CI-сервере другой часовой пояс (например, UTC)
- А лучше используйте в тестах специальные гемы типа timecop
- https://bugs.ruby-lang.org/issues/11261

HOWTO: Обмен данными

- Самое надёжное — формат ISO 8601 в UTC (он же RFC 3339):
  `'2015-05-18T22:16:38Z'`
- Если нужно именно локальное время, смещение обязательно:
  `'2015-05-19T01:16:38+03:00'`
- На клиенте, если у вас moment.js, то вам нужен метод toISOString()
- Angular.js сериализует в ISO 8601 по умолчанию.
- Соответственно нужно парсить на стороне сервера:
  ```
  Time.iso8601(params[:from]) rescue Time.parse(params[:from])
  ```
  (но я бы лучше вернул код ошибки 400, ей богу)

Важность смещения у локального времени

```
Time.zone.parse("2014-10-26T01:00:00")
# TZInfo::AmbiguousTime: 2014-10-26 01:00:00 is an
ambiguous local time.
Time.zone.parse("2014-10-26T01:00:00+04:00")
# => Sun, 26 Oct 2014 01:00:00 MSK +04:00
Time.zone.parse("2014-10-26T01:00:00+03:00")
# => Sun, 26 Oct 2014 01:00:00 MSK +03:00
Time.zone.parse("2014-10-26T01:00:00+04:00").utc
# => 2014-10-25 21:00:00 UTC
Time.zone.parse("2014-10-26T01:00:00+03:00").utc
# => 2014-10-25 22:00:00 UTC
```

Резюме

- Информация о прошедших и текущих событиях сохраняется и передаётся обычно только в UTC! (ISO8601 или Unix Timestamp)
- Для дат в будущем нужно думать. Серебряной пули нет.
- Что бы одно вы ни хранили — что-нибудь обязательно «съедет»
- В идеале нужно сохранять всё: [UTC, локальное время, часовой пояс]
- Если есть информация о часовом поясе — сохраняется его id из tzdata
- Если нет, а локальное время хранить нужно — сохраняете смещение.
- По возможности времени с клиента не верьте, всё делайте на сервере.
- Всегда держите на сервере настроенный NTP и самую последнюю версию гема tzinfo-data или системного пакета tzdata

MOAR!

- «Never say never» или Работаем с таймзонами правильно
  http://habrahabr.ru/company/mailru/blog/242645/
- The Problem with Time & Timezones
  https://youtu.be/-5wpm-gesOY
- Документация!
  http://api.rubyonrails.org/classes/ActiveSupport/TimeZone.html
  http://api.rubyonrails.org/classes/ActiveSupport/TimeWithZone.html
  http://api.rubyonrails.org/classes/Time.html

Всё!

- Ваши вопросы очень ценны — задавайте их.
- Багрепорты/багфиксы по всем упомянутым багам отправлены.
- А ещё у нас есть работа :-)

AT Consulting