# 【Active Record】基礎

# 環境編

#### ■環境構築編

- ▶ ☆ 0. Rubyのインストール
- ▶ ☆ 1. Active Record と SQLite用gem のインストール

## 基本編

#### ■初歩的注意

- ▶ ※ Active Record はORMの1種である。
- ▶ ※ Active Record は Rails の一部として開発されているので、そのガイド(日本語版)を参照されたい。また、Rails ドキュメントも参考になる。
- ▶ ※ 以降の構文はすべてRubyファイルのなかで使われるものである。

#### ■基本

- ▶ ☆ 準備 (テンプレ)
- ▶ 実際に発行しているSQL文を 標準出力させるように設定

#### ■表 foos に関する準備

- ▶ ※ DB中の表じたいはRubyプログラムではなくSQLなどでつくる。
- ▶ ※ 表名は foos のように必ずスネークケースかつ複数形でなければならない。
- ▶ ※ 表では <u>created\_at updated\_at</u> という名前の列をつくっておくと、作成日時、更新日時 を Active Record の方で自動で管理してくれる。
- ▶ 表 foos の録をオブとして扱えるように
- ▶ 録の値に制限かける
- ▶ 緑の値に複雑な制限

### コールバック

# 【Active Record】基礎

## 環境編

#### ■環境構築編

- ▶ ☆ 0. Rubyのインストール
- ▶ ☆ 1. Active Record と SQLite用gem のインストール

# 基本編

#### ■初歩的注意

- ▶ ※ Active Record はORMの1種である。
- ▶ ※ Active Record は Rails の一部として開発されているので、そのガイド(日本語版)を参照されたい。また、Rails ドキュメントも参考になる。
- ▶ ※ 以降の構文はすべてRubyファイルのなかで使われるものである。
- ▶ ※ ! で終わる名前のメソッドは「うまくいかなかった場合にエラーを起こす」という類のメソッドである。また、メソッド名の最後に ! をつけると、「うまくいかなかった場合にエラーを起こす」ように変更できることが多い。

#### ■基本

- ▶ ☆ 準備(テンプレ)
- ▶ 実際に発行しているSQL文を require 'logger'標準出力させるように設定 ActiveRecord::Base.logger = Logger.new(STDOUT)

#### ■表 foos に関する準備

- ▶ ※ DB中の表じたいはRubyプログラムではなくSQLなどでつくる。
- ▶ ※ 表名は foos のように必ずスネークケースかつ複数形でなければならない。
- ▶ ※ 表では <u>created\_at updated\_at</u> という名前の列をつくっておくと、作成日時、更新日時 を Active Record の方で自動で管理してくれる。
- ▶ 表 foos の録をオブとして扱えるように class Foo < ActiveRecord::Base』 end
- ▶ 録の値に制限かける Foo定義内に validates :attr1, ..., validationHelper1: value1, ...
- ▶ 録の値に複雑な制限 " validate do |foo| 」 ·· ↓ end

## コールバック

▶ ※ コールバックとは、表 foos の更新(録の追加、変更、削除)の前後に自動的に処理をさせる仕組みである。(トリガーに酷似)
▶ コールバック を設定
アソシエーション
▶ ※ アソシエーションは、表 foos と表 foobars を関連づけて扱えるようにする機能である。
▶ 表 foos と 表 foobars を関連付け
■表 foos の録を取得
▶ ※ 下記のほとんどのメソにおいて返り値は、録が見つかった場合 ActiveRecord::Relation オブ、見つからなかったり失敗したりした場合は空のそれである。例外には※で注釈した。なお、! をメソッド名の最後に加えれば、録が見つからなかったり失敗したりした場合にエラーを起こすことができる。
▶ 全列における <b>全録</b>
<b>▶ 特定の列</b> における全録
▶ ※以下の 🛕 は ℻ か ActiveRecord::Relation オブを表す。
▶ 最初・最後のn件に
▶ 主キーがある値の1件
▶ 列がある値の1件
▶ " (なければ録追加!)
, A/H
▶ 条件に合う複数件
<ul><li>★件に合う複数件</li><li>ある列で並び替え</li></ul>
▶ ある列で並び替え
<ul><li>▶ ある列で並び替え</li><li>▶ 最大n件の録のみに</li></ul>
<ul> <li>▶ ある列で並び替え</li> <li>▶ 最大n件の録のみに</li> <li>▶ m件目以降で "</li> <li>▶ 一連の取得処理にhogeと名付け</li> </ul>

- ▶ ※ コールバックとは、表 foos の更新(録の追加、変更、削除)の前後に自動的に処理をさせる仕組みである。(トリガーに酷似)
- ▶ コールバック Foo定義内に *callbackMethod*※:*methodName* ※ <u>before\_insert</u> 等を設定 protected ↓ def *methodName* ↓ · · ↓ end

### アソシエーション

- ▶ ※ アソシエーションは、表 foos と表 foobars を関連づけて扱えるようにする機能である。
- ▶ 表 foos と Foo定義内に associationMethod※¹:theOther※² ※¹ belongs\_to 等 表 foobars ※² :foobar か :foobars かは associationMethod で決まる ※対応する内容をFoobar定義内にも書く

#### ■表 foos の録を取得

- ▶ ※ 下記のほとんどのメソにおいて返り値は、録が見つかった場合 ActiveRecord::Relation オブ、見つからなかったり失敗したりした場合は空のそれである。例外には※で注釈した。なお、! をメソッド名の最後に加えれば、録が見つからなかったり失敗したりした場合にエラーを起こすことができる。
- ▶ 全列における**全録** Foo.all
- ▶ 特定の列における全録 Foo.select("列1, 列2, ...").all
- ▶ ※以下の △ は Foo か ActiveRecord::Relation オブを表す。
- ▶ 最初・最後のn件に △.first(n)・.last(n) ※ n 略で1件に ※: ARRオブ?; nil
- ▶ 主キーがある値の1件 △.find(primaryKey) ※: Fooオブ; エラー
- ▶ 列がある値の1件 △.find by(列: 値) ※合致した最初の1件 ※: Fooオブ; nil
- ▶ " (なければ**録追加**!) △.find or create by(列:値)
- ▶ 条件に合う複数件 △.where(条件的文字列) △.where(条件的書式文字列, 変数か値1, ...)△.where(列1: 条件1, ...) のどれか
- ▶ ある列で並び替え △.order("列1 desc, 列2, ...") や △.order(列1::desc, :列2, ...)
- ▶ 最大n件の録のみに △.limit(n)
- ▶ m件目以降で " △.limit(n).offset(m) または .limit(m, n)
- ▶ 一連の取得処理にhogeと名付け ・Foo定義内に def self.hoge(p1, ...) ↓ 処理 ↓ end △.hoge(arg1, ...)で呼出し可能に ・ scope :hoge, ->(p1, ...) { 処理 }
- ▶ 表 foobars, foobazes, ... との関連付け Foo.includes(:theOther1%, ...) を読み込ませた表 foos の録を取得 ※ :foobar か :foobars かは関連付け次第

#### ■表 foos を更新

▶ ※ ふつう、追加や更新の際にバリデーション(録の値の制限)を満たさなかった場合、エ

ラーにはならずただ単に追加や更新の処理がスキップされる。ただし、下記の追加や更新の

メソッド名の最後に 🧵 を加えれば、バリデーションに反する場合にエラーを起こすことが

▶ ※ ふつう、追加や更新の際にバリデーション(録の値の制限)を満たさなかった場合、エラーにはならずただ単に追加や更新の処理がスキップされる。ただし、下記の追加や更新のメソッド名の最後に 「を加えれば、バリデーションに反する場合にエラーを起こすことができる。

▶ 録を追加 Foo.create(attr1: value1, ...) ※: true ; false

▶ 録の値を変える Foo.update(primaryKey, attr1: value1, ...) や

△.update(attr1: value1, ...) ※: true; false ※巧遅

▶ 録の値に計算を加える △.update\_all("列 = 計算式") ※拙速

▶ 全録を削除 Foo.delete\_all