

【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 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 **created_at** **updated_at** 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
- ▶ 録の値に複雑な制限

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 **created_at** **updated_at** 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**全録**
- ▶ **特定の列**における全録
- ▶ ※ 以下の **△** は **Foo** 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!**)
- ▶ 条件に合う複数件
- ▶ ある列で並び替え
- ▶ 最大 n 件の録のみに
- ▶ m 件目以降で "
- ▶ 一連の取得処理にhogeと名付け△.hoge($arg1$, ...)で呼出し可能に
- ▶ 表 foobars, foobazes, ... との関連付けを読み込ませた表 foos の録を取得

■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

- ▶ ※ 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` ※ `before_insert` 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 と表 foobars を関連付け Foo定義内に `associationMethod※1 :theOther※2` ※¹ `belongs_to` 等 ※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, 変数カ値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, ... との関連付けを読み込ませた表 foos の録を取得 Foo.includes(:theOther1※, ...) ※ `: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できる。

- ▶ 録を追加
- ▶ 録の値を変える
- ▶ 録の値に計算を加える
- ▶ 全録を削除
- ▶ 条件に合う録を削除

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`
- ▶ 条件に合う録を削除 • `Foo.delete(primaryKey)` や △.delete_all ※拙速
◎`Foo.destroy(primaryKey)` や △.destroy_all ※巧遅