PowerEnum

Enumerations for Rails 3.X Done Right

Arthur Shagall
TMX Credit

How to represent an enumeration of values?

You have an attribute whose possible values are members of a relatively small set.

- primary colors - red, blue, yellow
- engine choice - I4, I4 +Turbo, V6, V8

What's needed:

- Scalable - what's OK with a thousand records won't cut the mustard with a billion.
- Flexible - other stakeholders, legacy DB's, etc. must be supported.
- Simple and easy to use
- Sane design - should do the Right Thing. Reduce technical debt, not contribute to it.

This sucks

Strings with model-level validations:

- Not scalable, space inefficient
- Error prone - soon enough, you have colors like "blue", "BLUE", "Blue", "bule", "blue with a hint of cyan".

This also sucks

Strings/Symbols map to integers in the db:

- Cryptic DB - can't tell what's what without looking at the code; Does color 1 mean "blue" or "red"?
- You can add lookup tables in the database, but then you're maintaining definitions in two places.

What I really want

My enums should be instances of an ActiveRecord model.

- Enums have semantic meaning
- Enums can have behavior
- Enum set is defined in one place (the DB)

Welcome to PowerEnum

```
# Do stuff like
Booking.create(:status => :pending)    # Create a Booking
book = Booking.with_status(:pending).first
book.status == BookingStatus[:pending] # True
book.status === :pending               # Also true

# Select a status
BookingStatus[:confirmed]  # Select by symbols
BookingStatus['confirmed'] # Or strings
BookingStatus[1]           # Or id's
# Or select multiple
BookingStatus[:confirmed, :pending, :rejected]
```

Usage: setup

Install Gem

- In Gemfile: gem 'power_enum'
- bundle install

Works in Rails 3.0, 3.1, 3.2

Works on Ruby 1.8.7, 1.9, JRuby >= 1.6

Database agnostic

Usage: create enum

```
rails generate enum BookingStatus
```

Gives you:

```
class CreateEnumBookingStatus < ActiveRecord::Migration

  def change
    create_enum :booking_status
  end

end

class BookingStatus < ActiveRecord::Base
  acts_as_enumerated
end
```

Usage: seed initial values

```
# There are safeguards built in so enum values don't
# accidentally change.
BookingStatus.update_enumerations_model do
  BookingStatus.create( :name => :confirmed )
  BookingStatus.create( :name => :pending )
  BookingStatus.create( :name => :withdrawn )
  BookingStatus.create( :name => :rejected )
end
```

Usage: class with an enumerated attribute

```
# Migration
create_table :bookings do |t|
  t.references :booking_statuses
  t.timestamps
end

# Class definition
class Booking < ActiveRecord::Base
  has_enumerated :status,
                 :class_name  => 'BookingStatus',
                 :foreign_key => 'booking_status_id'
end
```

Potential uses

With a bit of work, there are all sorts of interesting possibilities:

- Statuses
- Single Table Inheritance
- States in a Finite State Machine

Testing

Built in RSpec matchers:

- `foo.should act_as_enumerated`
- `Bar.should have_enumerated(:foo)`
- `foo.some_enum_attribute.should match_enum(:foo)`

Limitations

Number of enum values should be small:

- 100-200 items is a reasonable upper limit
- Good candidates
  - Age groups (0-21, 22-30, 31-40, 41+)
  - US States
  - Statuses
- Bad candidates
  - US Zip codes
  - City names

Enums are cached per-process

More Info

https://github.com/albertosaurus/power_enum
https://rubygems.org/gems/power_enum

Credit where credit is due: Massive thanks to Trevor Squires, Pivotal Labs, and all the fine folks at TMX Credit