

How to Upstream Your Code to Rails

Hartley McGuire



Hartley McGuire

@skipkayhil

Rails Triage Team

Senior Developer @ Shopify

How to Upstream Your Code to Rails

Why to Upstream Your Code to Rails

~~How~~ to Upstream Your Code to Rails

 Lower Maintenance Costs

 Improve Rails Knowledge

 Open Source Citizenship

 Collect Internet Points!



Lower Maintenance Costs



Improve Rails Knowledge

```
$ git blame <file>
```



```
$ git blame <file>
```

```
# ~/.gitconfig
```

```
[alias]
```

```
    context = blame
```

```
$ git context <file>
```

Code


Blame

688 lines (606 loc) · 25.3 KB


Older Newer



















6 years ago		Adding frozen_string_literal ...		1	# frozen_string_literal: true
				2	
7 years ago		applies new string literal con...		3	require "yaml"
				4	require "active_support/core_ext/hash/keys"
				5	require "active_support/core_ext/object/blank"
				6	require "active_support/key_generator"
7 months ago		Add Rails.application.messa...		7	require "active_support/message_verifiers"
6 months ago		Add Rails.application.deprec...		8	require "active_support/deprecation"
6 years ago		Add credentials using a gen...		9	require "active_support/encrypted_configuration"
5 years ago		Allow deprecated non-symb...		10	require "active_support/hash_with_indifferent_access"
4 years ago		Extract internal ActiveSuppo...		11	require "active_support/configuration_file"
6 years ago		[Railties] require_relative =>...		12	require "rails/engine"
				13	require "rails/secrets"
2 years ago		Delay loading Zeitwerk		14	require "rails/autoloaders"

<https://github.com/rails/rails/blame/main/railties/lib/rails/application.rb>



Code Blame 688 lines (606 loc) · 25.3 KB



















Older  Newer

6 years ago		Adding frozen_string_literal ...		1	# frozen_string_literal: true
				2	
7 years ago		applies new string literal con...		3	require "yaml"
				4	require "active_support/core_ext/hash/keys"
				5	require "active_support/core_ext/object/blank"
				6	require "active_support/key_generator"
7 months ago		Add Rails.application.messa...		7	require "active_support/message_verifiers"
6 months ago		Add Rails.application.deprec...		8	require "active_support/deprecation"
6 years ago		Add credentials using a gen...		9	require "active_support/encrypted_configuration"
5 years ago		Allow deprecated non-symb...		10	require "active_support/hash_with_indifferent_access"
4 years ago		Extract internal ActiveSuppo...		11	require "active_support/configuration_file"
6 years ago		[Railties] require_relative =>...		12	require "rails/engine"
				13	require "rails/secrets"
2 years ago		Delay loading Zeitwerk		14	require "rails/autoloaders"

<https://github.com/rails/rails/blame/main/railties/lib/rails/application.rb>

Code Blame 688 lines (606 loc) · 25.3 KB

Older Newer

6 years ago		Adding frozen_string_literal ...		1	# frozen_string_literal: true
				2	
7 years ago		applies new string literal con...		3	require "yaml"
				4	require "active_support/core_ext/hash/keys"
				5	require "active_support/core_ext/object/blank"
				6	require "active_support/key_generator"
7 months ago		Add Rails.application.messa...		7	require "active_support/message_verifiers"
6 months ago		Add Rails.application.deprec...		8	require "active_support/deprecation"
6 years ago		Add credentials using a gen...		9	require "active_support/encrypted_configuration"
5 years ago		Allow deprecated non-symb...		10	require "active_support/hash_with_indifferent_access"
4 years ago		Extract internal ActiveSuppo...		11	require "active_support/configuration_file"
6 years ago		[Railties] require_relative =>...		13	require "rails/engine"
					require "rails/secrets"
2 years ago		Delay loading Zeitwerk		14	require "rails/autoloaders"

<https://github.com/rails/rails/blame/main/railties/lib/rails/application.rb>

Code Blame 613 lines (537 loc) · 22.2 KB

Older

<

<https://github.com/rails/rails/blame/main/railties/lib/rails/application.rb>



Open Source Citizenship

 Collect Internet Points!



Rails Contributors - All time

Showing 6389 people

	Name	Since	Commits
#1	Rafael Mendonça França	06 Oct 2010	10741
#2	Aaron Patterson	10 Mar 2009	6464
#3	David Heinemeier Hansson	24 Nov 2004	4782
#4	Ryuta Kamizono	20 Nov 2013	4592
#5	Jeremy Daer	24 Nov 2004	4407
#6	Xavier Noria	14 Oct 2007	3022
#7	José Valim	14 May 2008	3014
#8	Eileen M. Uchitelle	31 Mar 2014	2130
#9	Yves Senn	20 May 2011	2069
#10	Santiago Pastorino	16 Jan 2010	1984

All time

Today

This week

This month

This year

Releases

Edge

FAQ

<https://contributors.rubyonrails.org>

Why to Upstream Your Code to Rails

~~How~~ to Upstream Your Code to Rails

What to Upstream to Rails

~~Why to Upstream Your Code to Rails~~

~~How to Upstream Your Code to Rails~~



Documentation



Bug Fixes



Performance Improvements



New features!



Documentation

Getting Started with Rails

This guide covers getting up and running with Ruby on Rails.

After reading this guide, you will know:

- ✔ How to install Rails, create a new Rails application, and connect your application to a database.
- ✔ The general layout of a Rails application.
- ✔ The basic principles of MVC (Model, View, Controller) and RESTful design.
- ✔ How to quickly generate the starting pieces of a Rails application.

1 Guide Assumptions

This guide is designed for beginners who want to get started with creating a Rails application from scratch. It does not assume that you have any prior experience with Rails.

Rails is a web application framework running on the Ruby programming language. If you have no prior experience with Ruby, you will find a very steep learning curve diving straight into Rails. There are several curated lists of online resources for learning Ruby:



Chapters

1. [Guide Assumptions](#)
2. [What is Rails?](#)
3. [Creating a New Rails Project](#)
 - [Installing Rails](#)
 - [Creating the Blog Application](#)
4. [Hello, Rails!](#)
 - [Starting up the Web Server](#)
 - [Say "Hello", Rails](#)
 - [Setting the Application Home Page](#)
5. [Autoloading](#)
6. [MVC and You](#)
 - [Generating a Model](#)
 - [Database Migrations](#)
 - [Using a Model to Interact with the Database](#)
 - [Showing a List of Articles](#)
7. [CRUDit Where CRUDit Is Due](#)

https://guides.rubyonrails.org/getting_started.html

Ruby on Rails Guides Guidelines

This guide documents guidelines for writing Ruby on Rails Guides. This guide follows itself in a graceful loop, serving itself as an example.

After reading this guide, you will know:

- ✔ **About the conventions to be used in Rails documentation.**
- ✔ **How to generate guides locally.**



Chapters

1. [Markdown](#)
2. [Prologue](#)
3. [Headings](#)
4. [Linking to the API](#)
5. [API Documentation Guidelines](#)
6. [HTML Guides](#)
 - [Generation](#)
 - [Validation](#)
7. [Kindle Guides](#)
 - [Generation](#)

1 Markdown

Guides are written in [GitHub Flavored Markdown](#). There is comprehensive [documentation for Markdown](#), as well as a [cheatsheet](#).

2 Prologue

Each guide should start with motivational text at the top (that's the little introduction in the blue area). The prologue should tell the reader what the guide is about, and what they will learn. As an example, see the [Routing Guide](#).

https://guides.rubyonrails.org/ruby_on_rails_guides_guidelines.html

Search (/) for a class, method, ...

files

Core extensions

AbstractController

ActionCable

ActionController

ActionDispatch

ActionMailbox

ActionMailer

ActionText

ActionView

ActiveJob

ActiveModel

ActiveRecord

ActiveStorage

▼ ActiveSupport

ActionableError

ArrayInquirer < Array

Autoload

BacktraceCleaner < Object

Benchmarkable

▶ Cache

CachingKeyGenerator < Object

▶ Callbacks

CodeGenerator < Object

CompareWithRange

Concern

▶ Concurrency

Ruby on Rails 7.0.4.2

Class

ActiveSupport::Duration < Object

activesupport/lib/active_support/duration.rb

activesupport/lib/active_support/duration/iso8601_parser.rb

activesupport/lib/active_support/duration/iso8601_serializer.rb

Provides accurate date and time measurements using `Date#advance` and `Time#advance`, respectively. It mainly supports the methods on `Numeric`.

```
1.month.ago # equivalent to Time.now.advance(months: -1)
```

Namespace

• CLASS ActiveSupport::Duration::ISO8601Parser

Methods

% , * , + , - , / , <=> , ==

A after, ago

B before, build

E eq?

F from_now

H hash

I in_days, in_hours, in_minutes, in_months, in_seconds, in_weeks, in_years, iso8601

<https://api.rubyonrails.org/classes/ActiveSupport/Duration.html>

API Documentation Guidelines

This guide documents the Ruby on Rails API documentation guidelines.

After reading this guide, you will know:

- ✔ **How to write effective prose for documentation purposes.**
- ✔ **Style guidelines for documenting different kinds of Ruby code.**

1 RDoc

The [Rails API documentation](#) is generated with [RDoc](#). To generate it, make sure you are in the rails root directory, run `bundle install` and execute:

```
$ bundle exec rake rdoc
```

[Copy](#)

Resulting HTML files can be found in the `./doc/rdoc` directory.

Please consult the RDoc documentation for help with the [markup](#), and also take into account these [additional directives](#).



Chapters



1. [RDoc](#)
2. [Links](#)
3. [Wording](#)
4. [English](#)
5. [Oxford Comma](#)
6. [Example Code](#)
7. [Booleans](#)
8. [File Names](#)
9. [Fonts](#)
 - [Fixed-width Font](#)
 - [Regular Font](#)
10. [Description Lists](#)
11. [Dynamically Generated Methods](#)
12. [Method Visibility](#)
13. [Regarding the Rails Stack](#)


https://guides.rubyonrails.org/api_documentation_guidelines.html





Bug Fixes


Fix collation for changing column to non-string #46400

 Merged byroot merged 1 commit into rails:7-0-stable from skipkayhil:backport-45807  on Nov 1, 2022

 Conversation 0

 Commits 1

 Checks 3

 Files changed 2



skipkayhil commented on Nov 1, 2022

Member



[3a0d0f4](#) made `change_column` preserve the existing column's collation. However, this leads to problems when the old and new column types have incompatible collations.

[be0a58b](#) addressed this by adding a check to ensure that the new column type isn't binary. However, this did not cover every column type impacted by this issue.

[008d9d1](#) followed up by changing the collation to only be preserved if both the old and new column types are string-like, fixing the issue for other column types like `:int` or `:blob`.

This is a manual backport of [008d9d1](#) because it also touched some other code only present on main.

Co-authored-by: Andrew Hoglund ahoglund@github.com

<https://github.com/rails/rails/pull/46400>

```
diff --git a/Gemfile b/Gemfile
index 45ae107e9..a92efe270 100644
--- a/Gemfile
+++ b/Gemfile
@@ -9,9 +9,9 @@
# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
-gem "rails", "7.0.4"
+gem "rails", github: "rails/rails", branch: "7-0-stable"
```



Performance Improvements

ActiveRecord: Improve `find_db_config` performance #47890

Merged jonathanhefner merged 1 commit into rails:main from alexcwatt:finddbconfig-perf 2 days ago

Conversation 4

Commits 1

Checks 8

Files changed 1



alexcwatt commented 3 days ago • edited ▾

Contributor ...

Motivation / Background

I noticed a large Rails app with many database configurations spending ~5ms on `find_db_config`, with about half of that time spent on `Array#<=>`. We can update `find_db_config` to avoid this expensive sort.

Detail

This PR changes `find_db_config` to not need sorting. Instead we do two `find` calls - the first time, looking at configs `for_current_env?` and the second time, considering all configs.

<https://github.com/rails/rails/pull/47890>

```
require "bundler/inline"
```

```
gemfile(true) do
```

```
  source "https://rubygems.org/"
```

```
  gem "rails"
```

```
  gem "benchmark-ips"
```

```
end
```

```
require "bundler/inline"

gemfile(true) do
  source "https://rubygems.org/"

  gem "rails"
  gem "benchmark-ips"
end
```

https://github.com/rails/rails/blob/main/guides/bug_report_templates/benchmark.rb

```
module ActiveRecord
  class DatabaseConfigurations
    def fast_find_db_config(env)
      # find_db_config but FASTER!
    end
  end
end
```



```
Benchmark.ips do |x|  
  x.report("find_db_config") { find_db_config("primary") }  
  x.report("fast_find_db_config") { fast_find_db_config("primary") }  
  x.compare!  
end
```

Comparison:

fast_find_db_config: 24073.0 i/s

find_db_config: 3621.5 i/s - 6.65x slower



New features!

What to Upstream to Rails

~~Why to Upstream Your Code to Rails~~

~~How to Upstream Your Code to Rails~~

~~What~~ to Upstream ~~to~~ Rails

~~Why~~ to Upstream ~~Your Code~~ to Rails

How to Upstream Your Code to Rails

```
class ImmutableValidator < ActiveRecord::EachValidator
```

```
end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
```



```
  def validate_each(record, attribute, _value)
```

```
    end
```

```
  end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
```

```
  def validate_each(record, attribute, _value)  
    #           #<Post ...>, :content, "Hello, world!"
```

```
  end
```

```
end
```

```
class Post < ActiveRecord::Base
```

```
  validates :content, immutable: true
```



```
end
```



```
class ImmutableValidator < ActiveRecord::EachValidator
```

```
  def validate_each(record, attribute, _value)
```

```
    if record.persisted?
```

```
      end
```

```
    end
```

```
  end
```

```
class Post < ActiveRecord::Base
```

```
  validates :content, immutable: true
```

```
end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
```

```
  def validate_each(record, attribute, _value)  
    if record.persisted? && record.attribute_changed?(attribute)
```



```
  end  
end  
end
```

```
class Post < ActiveRecord::Base  
  validates :content, immutable: true  
end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
```

```
  def validate_each(record, attribute, _value)
```

```
    if record.persisted? && record.attribute_changed?(attribute)
```

```
      record.errors.add(attribute, "is immutable")
```

```
    end
```

```
  end
```

```
end
```

```
class Post < ActiveRecord::Base
```

```
  validates :content, immutable: true
```

```
end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
```

```
  def validate_each(record, attribute, _value)
    if record.persisted? && record.attribute_changed?(attribute)
      record.errors.add(attribute, "is immutable")
    end
  end
end
```

```
class Post < ActiveRecord::Base
  belongs_to :author
  validates :author, immutable: true
end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
```

```
➡ def attribute_name(record, attribute)
```

```
end
```

```
def validate_each(record, attribute, _value)
```

```
  if record.persisted? && record.attribute_changed?(attribute_name(record, attribute))
```

```
    record.errors.add(attribute, "is immutable")
```

```
  end
```

```
end
```


```
end
```

```
class Post < ActiveRecord::Base
```

```
  belongs_to :author
```

```
  validates :author, immutable: true
```

```
end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
  def attribute_name(record, attribute)
     return attribute if record.attributes.include?(attribute.to_s)
  end

  def validate_each(record, attribute, _value)
    if record.persisted? && record.attribute_changed?(attribute_name(record, attribute))
      record.errors.add(attribute, "is immutable")
    end
  end
end

class Post < ActiveRecord::Base
  belongs_to :author
  validates :author, immutable: true
end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
  def attribute_name(record, attribute)
    return attribute if record.attributes.include?(attribute.to_s)
    record.association(attribute.to_s).reflection.foreign_key
  end

  def validate_each(record, attribute, _value)
    if record.persisted? && record.attribute_changed?(attribute_name(record, attribute))
      record.errors.add(attribute, "is immutable")
    end
  end
end

class Post < ActiveRecord::Base
  belongs_to :author
  validates :author, immutable: true
end
```

```
class ImmutableValidator < ActiveRecord::EachValidator
  def attribute_name(record, attribute)
    return attribute if record.attributes.include?(attribute.to_s)

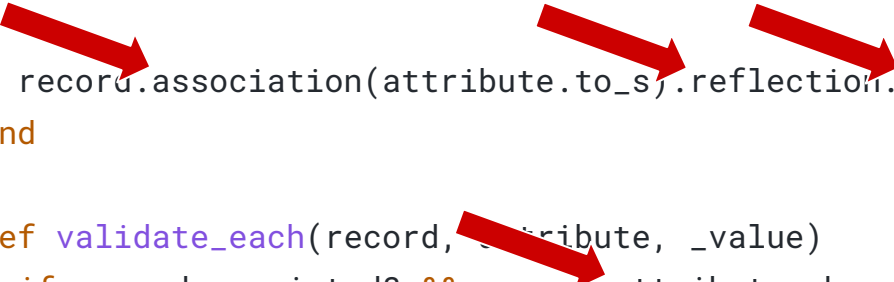
    record.association(attribute.to_s).reflection.foreign_key
  end

  def validate_each(record, attribute, _value)
    if record.persisted? && record.attribute_changed?(attribute_name(record, attribute))
      record.errors.add(attribute, "is immutable")
    end
  end
end
```



```
class ImmutableValidator < ActiveRecord::EachValidator
  def attribute_name(record, attribute)
    return attribute if record.attributes.include?(attribute.to_s)
    record.association(attribute.to_s).reflection.foreign_key
  end

  def validate_each(record, attribute, _value)
    if record.persisted? && record.attribute_changed?(attribute_name(record, attribute))
      record.errors.add(attribute, "is immutable")
    end
  end
end
```



```
class ImmutableValidator < ActiveRecord::EachValidator
  def attribute_name(record, attribute)
    return attribute if record.attributes.include?(attribute.to_s)

    record.association(attribute.to_s).reflection.foreign_key
  end

  def validate_each(record, attribute, _value)
    if record.persisted? && record.attribute_changed?(attribute_name(record, attribute))
      record.errors.add(attribute, "is immutable")
    end
  end
end
```

attr_readonly(*attributes)

[Link](#)

Attributes listed as readonly will be used to create a new record but update operations will ignore these fields.

You can assign a new value to a readonly attribute, but it will be ignored when the record is updated.

Examples

```
class Post < ActiveRecord::Base
  attr_readonly :title
end

post = Post.create!(title: "Introducing Ruby on Rails!")
post.update(title: "a different title") # change to title will be ignored
```

Source: [show](#) | [on GitHub](#)

https://api.rubyonrails.org/classes/ActiveRecord/ReadonlyAttributes/ClassMethods.html#method-i-attr_readonly

```
class ImmutableValidator < ActiveRecord::EachValidator
  def attribute_name(record, attribute)
    return attribute if record.attributes.include?(attribute.to_s)

    record.association(attribute.to_s).reflection.foreign_key
  end

  def validate_each(record, attribute, _value)
    if record.persisted? && record.attribute_changed?(attribute_name(record, attribute))
      record.errors.add(attribute, "is immutable")
    end
  end
end
```

```
module ActiveRecord
  module Validations
    class ReadonlyValidator < ActiveModel::EachValidator
      def validate_each(record, attribute, _value)
        if record.persisted? && record.attribute_changed?(attribute_name(record, attribute))
          record.errors.add(attribute, "is immutable")
        end
      end

      private

      def attribute_name(record, attribute)
        return attribute if record.attributes.include?(attribute.to_s)

        record.association(attribute.to_s).reflection.foreign_key
      end
    end
  end
end
```

```
$ git add . && git commit -m "Introduce ReadonlyValidator"
```

<!--

Thanks for contributing to Rails!

Please do not make **Draft** pull requests, as they still send notifications to everyone watching the Rails repo.

Create a pull request when it is ready for review and feedback from the Rails team :).

If your pull request affects documentation or any non-code changes, guidelines for those changes are [available here](https://edgeguides.rubyonrails.org/contributing_to_ruby_on_rails.html#contributing-to-the-rails-documentation)

About this template

The following template aims to help contributors write a good description for their pull requests.

We'd like you to provide a description of the changes in your pull request (i.e. bugs fixed or features added), motivation behind the changes, and complete the checklist below before opening a pull request.

Feel free to discard it if you need to (e.g. when you just fix a typo). -->

Motivation / Background

<!--

Describe why this Pull Request needs to be merged. What bug have you fixed? What feature have you added? Why is it important?

If you are fixing a specific issue, include "Fixes #ISSUE" (replace with the issue number, remove the quotes) and the issue will be linked to this PR.

-->

This Pull Request has been created because [REPLACE ME]

Motivation / Background

<!--

Describe why this Pull Request needs to be merged. What bug have you fixed? What feature have you added? Why is it important?

If you are fixing a specific issue, include "Fixes #ISSUE" (replace with the issue number, remove the quotes) and the issue will be linked to this PR.

-->

This Pull Request has been created because [REPLACE ME]

Motivation / Background

`attr_readonly` prevents attribute changes from being persisted to the database, however it has a downside that this happens silently.

Motivation / Background

attr_readonly prevents attribute changes from being persisted to the database, however it has a downside that this happens silently.

Detail

This Pull Request changes [REPLACE ME]

Motivation / Background

`attr_readonly` prevents attribute changes from being persisted to the database, however it has a downside that this happens silently.

Detail

`ReadonlyValidator` is an alternative to `attr_readonly`, which uses `ActiveModel::Validations` to throw a validation error when something tries to update a readonly attribute

```
```\n\nclass Reply\n  belongs_to :topic\n\n  validates :topic, readonly: true\nend\n```\n
```

Co-authored-by: Jeremy Cole <[jeremy.cole@shopify.com](mailto:jeremy.cole@shopify.com)>

```
$ git commit --amend
```

## Introduce ReadonlyValidator

`attr_readonly` prevents attribute changes from being persisted to the database, however it has a downside that this happens silently.

`ReadonlyValidator` is an alternative to `attr_readonly`, which uses `ActiveModel::Validations` to throw a validation error when something tries to update a readonly attribute

```
...
```

```
class Reply
 belongs_to :topic

 validates :topic, readonly: true
end
...
```

Co-authored-by: Jeremy Cole <jeremy.cole@shopify.com>

### Additional information

<!-- Provide additional information such as benchmarks, reference to other repositories or alternative solutions. -->

### Additional information

### Checklist

Before submitting the PR make sure the following are checked:

### Additional information

### Checklist

Before submitting the PR make sure the following are checked:

\* ☐ This Pull Request is related to one change.



### Additional information

### Checklist

Before submitting the PR make sure the following are checked:

- \* ☒ This Pull Request is related to one change.

### Additional information

### Checklist

Before submitting the PR make sure the following are checked:

- \* ☒ This Pull Request is related to one change.
- \* ☐ Commit message has a detailed description of what changed and why.

### Additional information

### ### Checklist

Before submitting the PR make sure the following are checked:

- \* [X] This Pull Request is related to one change.
- \* [X] Commit message has a detailed description of what changed and why.

### Additional information

### Checklist

Before submitting the PR make sure the following are checked:

- \* ☒ This Pull Request is related to one change.
- \* ☒ Commit message has a detailed description of what changed and why.
- \* ☐ Tests are added or updated if you fix a bug or add a feature.

```
$ git commit --amend
```

```
class ReadonlyValidationTest < ActiveRecord::TestCase
 fixtures :topics

 repair_validations(Topic)

 test "readonly attributes can be set on creation" do
 Topic.validates :title, readonly: true
 t = Topic.new(title: "Ruby")
 assert_predicate t, :valid?
 end

 test "readonly attributes can be updated before persisting" do
 Topic.validates :title, readonly: true
 t = Topic.new(title: "Ruby")
 t.update(title: "Ruby is great!")
 assert_predicate t, :valid?
 end

 test "validates persisted attribute can not be changed" do
 Topic.validates :title, readonly: true
 t = Topic.create!(title: "Ruby")

 error = assert_raises(ActiveRecord::RecordInvalid) do
 t.update!(title: "Ruby is great!")
 end

 assert_equal("Validation failed: Title is readonly", error.message)
 end

 test "validates persisted association can not be changed" do
 Reply.validates :topic, readonly: true

 r = Reply.create!(title: "Yes it is!")

 Topic.create!(title: "Ruby is great!", replies: [r])

 error = assert_raises(ActiveRecord::RecordInvalid) do
 r.update!(topic: Topic.create!(title: "Rails is great!"))
 end
 assert_equal("Validation failed: Topic is readonly", error.message)
 end
end
```

### ### Additional information

### ### Checklist

Before submitting the PR make sure the following are checked:

- \* [X] This Pull Request is related to one change.
- \* [X] Commit message has a detailed description of what changed and why.
- \* [X] Tests are added or updated if you fix a bug or add a feature.

### Additional information

### ### Checklist

Before submitting the PR make sure the following are checked:

- \* ☒ This Pull Request is related to one change.
- \* ☒ Commit message has a detailed description of what changed and why.
- \* ☒ Tests are added or updated if you fix a bug or add a feature.
- \* ☐ CHANGELOG files are updated for the changed libraries.

```
$ git commit --amend
```

## \* Introduce ReadonlyValidator

An alternative to `attr_readonly`, which uses `ActiveModel::Validations` to throw a validation error when something tries to update a readonly attribute.

```
```\n\nclass Reply\n  belongs_to :topic\n  validates :topic, readonly: true\nend\n```\n
```

Hartley McGuire, *Jeremy Cole*

Additional information

Checklist

Before submitting the PR make sure the following are checked:

- * [X] This Pull Request is related to one change.
- * [X] Commit message has a detailed description of what changed and why.
- * [X] Tests are added or updated if you fix a bug or add a feature.
- * [X] CHANGELOG files are updated for the changed libraries.



hmcguire-shopify commented on Sep 21, 2022

Contributor



Motivation / Background

`attr_readonly` prevents attribute changes from being persisted to the database, however it has a downside that this happens silently.

Detail

`ReadonlyValidator` is an alternative to `attr_readonly`, which uses `ActiveModel::Validations` to throw a validation error when something tries to update a readonly attribute

```
class Reply
  belongs_to :topic

  validates :topic, readonly: true
end
```



Co-authored-by: Jeremy Cole jeremy.cole@shopify.com

<https://github.com/rails/rails/pull/46092>



rafaelfranca commented on Sep 21, 2022



Member ...


Validation are for user input. A read only attribute being set by user input is a developer error, so I don't think a validation can be used to this. Although, I can see we changing `attr_readonly` to raise an exception when the field is assigned. Isn't that possible to implement?





<https://github.com/rails/rails/pull/46092>


Make assigning a readonly attribute a no-op #42705

 Closed ghiculescu wants to merge 1 commit into rails:main from ghiculescu:readonly-assign 

 Conversation 18

 Commits 1

 Checks 3

 Files changed 3



ghiculescu commented on Jul 5, 2021

Member ...

Fixes #41520

After thinking about this a bit, I think this is a bug and the behaviour should match saving, for non-new records.



<https://github.com/rails/rails/pull/42705>

```

diff --git a/activerecord/lib/active_record/readonly_attributes.rb
b/activerecord/lib/active_record/readonly_attributes.rb
index c98d1528c698..0083e5fffc24 100644
--- a/activerecord/lib/active_record/readonly_attributes.rb
+++ b/activerecord/lib/active_record/readonly_attributes.rb
@@ -23,7 +23,15 @@ module ClassMethods
  #   post = Post.create!(title: "Introducing Ruby on Rails!")
  #   post.update(title: "a different title") # change to title will be ignored
  def attr_readonly(*attributes)
-   self._attr_readonly = Set.new(attributes.map(&:to_s)) + (_attr_readonly || [])
+   new_attributes = attributes.map(&:to_s).reject { |a| _attr_readonly.include?(a) }
+
+   new_attributes.each do |attribute|
+     define_method("#{attribute}=") do |value|
+       super(value) if new_record?
+     end
+   end
+
+   self._attr_readonly = Set.new(new_attributes) + _attr_readonly
  end

  # Returns an array of all the attributes that have been specified as readonly.


```

```
diff --git a/activerecord/lib/active_record/readonly_attributes.rb
b/activerecord/lib/active_record/readonly_attributes.rb
index c98d1528c698..0083e5fffc24 100644
--- a/activerecord/lib/active_record/readonly_attributes.rb
+++ b/activerecord/lib/active_record/readonly_attributes.rb
@@ -23,7 +23,15 @@ module ClassMethods
  #   post = Post.create!(title: "Introducing Ruby on Rails!")
  #   post.update(title: "a different title") # change to title will be ignored
+  def attr_readonly(*attributes)
-    self._attr_readonly = Set.new(attributes.map(&:to_s)) + (_attr_readonly || [])
+    new_attributes = attributes.map(&:to_s).reject { |a| _attr_readonly.include?(a) }
+
+    new_attributes.each do |attribute|
+      define_method("#{attribute}=") do |value|
+        super(value) if new_record?
+      end
+    end
+
+    self._attr_readonly = Set.new(new_attributes) + _attr_readonly
  end

  # Returns an array of all the attributes that have been specified as readonly.
```

```
diff --git a/activerecord/lib/active_record/readonly_attributes.rb
b/activerecord/lib/active_record/readonly_attributes.rb
index c98d1528c698..0083e5fffc24 100644
--- a/activerecord/lib/active_record/readonly_attributes.rb
+++ b/activerecord/lib/active_record/readonly_attributes.rb
@@ -23,7 +23,15 @@ module ClassMethods
  #   post = Post.create!(title: "Introducing Ruby on Rails!")
  #   post.update(title: "a different title") # change to title will be ignored
  def attr_readonly(*attributes)
-   self._attr_readonly = Set.new(attributes.map(&:to_s)) + (_attr_readonly || [])
+   new_attributes = attributes.map(&:to_s).reject { |a| _attr_readonly.include?(a) }
+
+   new_attributes.each do |attribute|
+     define_method("#{attribute}=") do |value|
+       super(value) if new_record?
+     end
+   end
+
+   self._attr_readonly = Set.new(new_attributes) + _attr_readonly
  end

  # Returns an array of all the attributes that have been specified as readonly.
```




```


@@ -23,7 +26,21 @@ module ClassMethods
  #   post = Post.create!(title: "Introducing Ruby on Rails!")
  #   post.update(title: "a different title") # change to title will be ignored
  def attr_readonly(*attributes)
-     self._attr_readonly = Set.new(attributes.map(&:to_s)) + (_attr_readonly || [])
+     new_attributes = attributes.map(&:to_s).reject { |a| _attr_readonly.include?(a) }
+
+     if ActiveRecord.raise_on_assign_to_attr_readonly
+       new_attributes.each do |attribute|
+         define_method("#{attribute}=") do |value|
+           raise ReadonlyAttributeError.new(attribute) unless new_record?
+
+           super(value)
+         end
+       end
+
+       include(HasReadonlyAttributes)
+     end
+
+     self._attr_readonly = Set.new(new_attributes) + _attr_readonly
  end

```




```
@@ -23,7 +26,21 @@ module ClassMethods
  #   post = Post.create!(title: "Introducing Ruby on Rails!")
  #   post.update(title: "a different title") # change to title will be ignored
  def attr_readonly(*attributes)
-     self._attr_readonly = Set.new(attributes.map(&:to_s)) + (_attr_readonly || [])
+     new_attributes = attributes.map(&:to_s).reject { |a| _attr_readonly.include?(a) }
+
+     if ActiveRecord.raise_on_assign_to_attr_readonly
+       new_attributes.each do |attribute|
+         define_method("#{attribute}=") do |value|
+           raise ReadonlyAttributeError.new(attribute) unless new_record?
+
+           super(value)
+         end
+       end
+
+       include(HasReadonlyAttributes)
+     end
+
+     self._attr_readonly = Set.new(new_attributes) + _attr_readonly
  end
end
```



```
@@ -23,7 +26,21 @@ module ClassMethods
  #   post = Post.create!(title: "Introducing Ruby on Rails!")
  #   post.update(title: "a different title") # change to title will be ignored
  def attr_readonly(*attributes)
-     self._attr_readonly = Set.new(attributes.map(&:to_s)) + (_attr_readonly || [])
+     new_attributes = attributes.map(&:to_s).reject { |a| _attr_readonly.include?(a) }
+
+     if ActiveRecord.raise_on_assign_to_attr_readonly
+       new_attributes.each do |attribute|
+         define_method("#{attribute}=") do |value|
+            raise ReadonlyAttributeError.new(attribute) unless new_record?
+         end
+       end
+     end
+
+     include(HasReadonlyAttributes)
+   end
+
+   self._attr_readonly = Set.new(new_attributes) + _attr_readonly
  end
```

```
@@ -23,7 +26,21 @@ module ClassMethods
  #   post = Post.create!(title: "Introducing Ruby on Rails!")
  #   post.update(title: "a different title") # change to title will be ignored
  def attr_readonly(*attributes)
-     self._attr_readonly = Set.new(attributes.map(&:to_s)) + (_attr_readonly || [])
+     new_attributes = attributes.map(&:to_s).reject { |a| _attr_readonly.include?(a) }
+
+     if ActiveRecord.raise_on_assign_to_attr_readonly
+       new_attributes.each do |attribute|
+         define_method("#{attribute}=") do |value|
+           raise ReadonlyAttributeError.new(attribute) unless new_record?
+
+           super(value)
+         end
+       end
+     end
+
+     include(HasReadonlyAttributes)
+
+     self._attr_readonly = Set.new(new_attributes) + _attr_readonly
  end
end
```



```
@@ -35,5 +52,15 @@ def readonly_attribute?(name) # :nodoc:
    _attr_readonly.include?(name)
  end
end

+
+ module HasReadonlyAttributes # :nodoc:
+   def write_attribute(attr_name, value)
+     if !new_record? && self.class.readonly_attribute?(attr_name.to_s)
+       raise ReadonlyAttributeError.new(attr_name)
+     end
+
+     super
+   end
+ end
end
end
```

```
@@ -35,5 +52,15 @@ def readonly_attribute?(name) # :nodoc:
```

```
  _attr_readonly.include?(name)
```

```
end
```

```
end
```

```
+
```

```
+ module HasReadonlyAttributes # :nodoc:
```

```
+   def write_attribute(attr_name, value)
```

```
+     if !new_record? && self.class.readonly_attribute?(attr_name.to_s)
```

```
+       raise ReadonlyAttributeError.new(attr_name)
```

```
+     end
```

```
+
```

```
+     super
```

```
+   end
```

```
+ end
```

```
end
```

```
end
```



```
post.content = "New content"
```

```
post[:content] = "New content"
```

```
post.assign_attributes(content: "New content")
```

```
post.update(content: "New content")
```

```
post.write_attribute(:content, "New content")
```

```
@@ -35,5 +52,15 @@ def readonly_attribute?(name) # :nodoc:
```

```
  _attr_readonly.include?(name)
```

```
  end
```

```
end
```

```
+
```

```
module HasReadonlyAttributes # :nodoc:
```

```
+ def write_attribute(attr_name, value)
```

```
+   if !new_record? && self.class.readonly_attribute?(attr_name.to_s)
```

```
+     raise ReadonlyAttributeError.new(attr_name)
```

```
+   end
```

```
+
```

```
+   super
```

```
+ end
```

```
+ end
```

```
end
```

```
end
```

```
post.content = "New content"
```

```
post[:content] = "New content"
```

```
post.assign_attributes(content: "New content")
```

```
post.update(content: "New content")
```

```
post.write_attribute(:content, "New content")
```

```

00 -691,16 +699,132 00 def test_comparison_with_different_objects_in_array
end

def test_readonly_attributes
  - assert_equal Set.new([ "title", "comments_count" ]), ReadonlyTitlePost.readonly_attributes
+ assert_equal Set.new([ "title" ]), ReadonlyTitlePost.readonly_attributes

  post = ReadonlyTitlePost.create(title: "cannot change this", body: "changeable")
+ assert_equal "cannot change this", post.title
+ assert_equal "changeable", post.body
+
+ post = Post.find(post.id)
+ assert_equal "cannot change this", post.title
+ assert_equal "changeable", post.body
+
+ assert_raises(ActiveRecord::ReadonlyAttributeError) do
+   post.title = "changed via assignment"
+ end
+ post.body = "changed via assignment"
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via assignment", post.body
+
+ assert_raises(ActiveRecord::ReadonlyAttributeError) do
+   post.write_attribute(:title, "changed via write_attribute")
+ end
+ post.write_attribute(:body, "changed via write_attribute")
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via write_attribute", post.body
+
+ assert_raises(ActiveRecord::ReadonlyAttributeError) do
+   post.assign_attributes(body: "changed via assign_attributes", title: "changed via assign_attributes")
+ end
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via assign_attributes", post.body
+
+ assert_raises(ActiveRecord::ReadonlyAttributeError) do
+   post.update(title: "changed via update", body: "changed via update")
+ end
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via assign_attributes", post.body
+
+ assert_raises(ActiveRecord::ReadonlyAttributeError) do
+   post[:title] = "changed via []="
+ end
+ post[:body] = "changed via []="
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via []=", post.body
+
+ post.save!
+
+ post = Post.find(post.id)
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via []=", post.body
+ end

def test_readonly_attributes_on_a_new_record
+ assert_equal Set.new([ "title" ]), ReadonlyTitlePost.readonly_attributes
+
+ post = ReadonlyTitlePost.new(title: "can change this until you save", body: "changeable")
+ assert_equal "can change this until you save", post.title
+ assert_equal "changeable", post.body
+
+ post.title = "changed via assignment"
+ post.body = "changed via assignment"
+ assert_equal "changed via assignment", post.title
+ assert_equal "changed via assignment", post.body
+
+ post.write_attribute(:title, "changed via write_attribute")
+ post.write_attribute(:body, "changed via write_attribute")
+ assert_equal "changed via write_attribute", post.title
+ assert_equal "changed via write_attribute", post.body

```

```

+ post.assign_attributes(body: "changed via assign_attributes", title: "changed via assign_attributes")
+ assert_equal "changed via assign_attributes", post.title
+ assert_equal "changed via assign_attributes", post.body
+
+ post[:title] = "changed via []="
+ post[:body] = "changed via []="
+ assert_equal "changed via []=", post.title
+ assert_equal "changed via []=", post.body
+
+ post.save!
+
+ post = Post.find(post.id)
+ assert_equal "changed via []=", post.title
+ assert_equal "changed via []=", post.body
+ end

def test_readonly_attributes_when_configured_to_not_raise
+ assert_equal Set.new([ "title" ]), NonRaisingPost.readonly_attributes
+
+ post = NonRaisingPost.create(title: "cannot change this", body: "changeable")
+ assert_equal "cannot change this", post.title
+ assert_equal "changeable", post.body
+
+ post = Post.find(post.id)
+ assert_equal "cannot change this", post.title
+ assert_equal "changeable", post.body
+
+ post.title = "changed via assignment"
+ post.body = "changed via assignment"
+ post.save!
+ post.reload
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via assignment", post.body
+
+ post.write_attribute(:title, "changed via write_attribute")
+ post.write_attribute(:body, "changed via write_attribute")
+ post.save!
+ post.reload
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via write_attribute", post.body
+
+ post.assign_attributes(body: "changed via assign_attributes", title: "changed via assign_attributes")
+ post.save!
+ post.reload
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via assign_attributes", post.body
+
+ post.update(title: "changed via update", body: "changed via update")
+ post.reload
+ assert_equal "cannot change this", post.title
+ assert_equal "changed via update", post.body
+
+ post.update(title: "try to change", body: "changed")
+ post[:title] = "changed via []="
+ post[:body] = "changed via []="
+ post.save!
+ post.reload
+ assert_equal "cannot change this", post.title
+ - assert_equal "changed", post.body
+ assert_equal "changed via []=", post.body
+ end

def test_unicode_column_name

```

* Raise on assignment to readonly attributes

```
```ruby
class Post < ActiveRecord::Base
 attr_readonly :content
end
Post.create!(content: "cannot be updated")
post.content # "cannot be updated"
post.content = "something else" # => ActiveRecord::ReadOnlyAttributeError
```
```

Previously, assignment would succeed but silently not write to the database.

This behavior can be controlled by configuration:

```
```ruby
config.active_record.raise_on_assign_to_attr_readonly = true
```
```

and will be enabled by default with ``load_defaults 7.1``

Alex Ghiculescu*, *Hartley McGuire


```
$ git add . && git commit
```

Raise on assignment to readonly attributes

Previously, assignment would succeed but silently not write to the database.

The changes to counter_cache are necessary because incrementing the counter cache for a column calls []. I investigated an approach to use _write_attribute instead, however counter caches are expected to resolve attribute aliases so write_attribute/[]= seems more correct.

Similarly, []= was replaced with _write_attribute in merge_target_lists to skip the overridden []= and the primary key check. attribute_names will already return custom primary keys so the primary_key check in write_attribute is not needed.

Co-authored-by: Alex Ghiculescu <alex@tanda.co>

Additional information

<!-- Provide additional information such as benchmarks, reference to other repositories or alternative solutions. -->

Additional information

Based on [#42705](#)

Closes [#46092](#)

Additional information

Based on [#42705](#)

Closes [#46092](#)

Checklist

Before submitting the PR make sure the following are checked:

- * ☐ This Pull Request is related to one change.
- * ☐ Commit message has a detailed description of what changed and why.
- * ☐ Tests are added or updated if you fix a bug or add a feature.
- * ☐ CHANGELOG files are updated for the changed libraries.

Additional information

Based on [#42705](#)

Closes [#46092](#)

Checklist

Before submitting the PR make sure the following are checked:

- * [X] This Pull Request is related to one change.
- * [X] Commit message has a detailed description of what changed and why.
- * [X] Tests are added or updated if you fix a bug or add a feature.
- * [X] CHANGELOG files are updated for the changed libraries.



hmcguire-shopify commented on Sep 22, 2022 • edited ▾

Contributor



Motivation / Background

Previously, assignment would succeed but silently not write to the database.

Detail

The changes to `counter_cache` are necessary because incrementing the counter cache for a column calls `[]=`. I investigated an approach to use `_write_attribute` instead, however counter caches are expected to resolve attribute aliases so `write_attribute/[]=` seems more correct.

Similarly, `[]=` was replaced with `_write_attribute` in `merge_target_lists` to skip the overridden `[]=` and the primary key check. `attribute_names` will already return custom primary keys so the `primary_key` check in `write_attribute` is not needed.

Co-authored-by: Alex Ghiculescu alex@tanda.co

Additional information

Based on [#42705](#)

Closes [#46092](#)

<https://github.com/rails/rails/pull/46105>

Rails Discord server is now open to the public

Posted by rafaelfranca

In the past few weeks, the Rails team has been working on a few changes to lower the barrier for new contributors to the framework.

We started by **disabling the automatic closing of stale PRs**. While initially this automation helped the project keep the issues tracker under control, nowadays things are better and we prefer to go back to full human interaction.

To make it easier for contributors to recognize all the members of the Rails team, we made public the **list of current members of the Issues team**. The Issues team assists with issues triage, pull request reviews and documentation improvements, so they are often the first interaction users have with the Rails team.

Today, we are opening a Discord server to allow contributors to help each other and lower the overhead of communicating with new contributors.

<https://rubyonrails.org/2022/6/13/rails-discord-server-is-now-open-to-the-public>

Raise on assignment to readonly attributes #46105

Merged

rafaelfranca merged 1 commit into rails:main from hmcguire-shopify:raise-readonly-writer on Nov 17, 2022



Conversation 2



Commits 1



Checks 3



Files changed 14



hmcguire-shopify commented on Sep 22, 2022 • edited

Contributor



Motivation / Background

Previously, assignment would succeed but silently not write to the database.

Detail

The changes to counter_cache are necessary because incrementing the counter cache for a column calls []. I investigated an approach to use _write_attribute instead, however counter caches are expected to resolve attribute aliases so write_attribute/[]= seems more correct.

Similarly, []= was replaced with _write_attribute in merge_target_lists to skip the overridden []= and the primary key check. attribute_names will already return custom primary keys so the primary_key check in write_attribute is not needed.

Co-authored-by: Alex Ghiculescu alex@tanda.co

Additional information


Based on [#42705](#)

Closes [#46092](#)

<https://github.com/rails/rails/pull/46105>



```
NoMethodError: super: no superclass method `content=' for #<PostSubclass ...>  
Did you mean?  content  
               content?
```

```
@@ -23,7 +26,21 @@ module ClassMethods
  #   post = Post.create!(title: "Introducing Ruby on Rails!")
  #   post.update(title: "a different title") # change to title will be ignored
  def attr_readonly(*attributes)
-     self._attr_readonly = Set.new(attributes.map(&:to_s)) + (_attr_readonly || [])
+     new_attributes = attributes.map(&:to_s).reject { |a| _attr_readonly.include?(a) }
+
+     if ActiveRecord.raise_on_assign_to_attr_readonly
+       new_attributes.each do |attribute|
+         define_method("#{attribute}=") do |value|
+           raise ReadonlyAttributeError.new(attribute) unless new_record?
+
+           super(value)
+         end
+       end
+
+       include(HasReadonlyAttributes)
+     end
+
+     self._attr_readonly = Set.new(new_attributes) + _attr_readonly
  end
end
```

```
diff --git a/activerecord/lib/active_record/readonly_attributes.rb b/activerecord/lib/active_record/readonly_attributes.rb
index d38d2fe0023bf..310d105a765ab 100644
--- a/activerecord/lib/active_record/readonly_attributes.rb
+++ b/activerecord/lib/active_record/readonly_attributes.rb
@@ -46,14 +46,20 @@ def readonly_attribute?(name) # :nodoc:
   end

   module HasReadOnlyAttributes # :nodoc:
     def write_attribute(attr_name, value)
       if !new_record? && self.class.readonly_attribute?(attr_name.to_s)
         raise ReadOnlyAttributeError.new(attr_name)
       end

       super
     end

+    def _write_attribute(attr_name, value)
+      if !new_record? && self.class.readonly_attribute?(attr_name.to_s)
+        raise ReadOnlyAttributeError.new(attr_name)
+      end
+
+      super
+    end
   end
end
```

```
class ReadonlyTitlePostSubclass < ReadonlyTitlePost
end

def test_classes_with_attr_readonly_can_be_subclassed
  post = ReadonlyTitlePostSubclass.create!

  assert_raises(ReadonlyAttributeError) do
    post.title = "Set via Create"
  end
end
```

```
class ReadonlyTitleAbstractPost < ActiveRecord::Base
```

```
  self.abstract_class = true
```

```
  attr_readonly :title
```

```
end
```

```
class ReadonlyTitlePostWithAbstractParent < ReadonlyTitleAbstractPost
```

```
  self.table_name = "posts"
```

```
end
```

```
def test_readonly_attributes_in_abstract_class_descendant
```

```
  assert_equal [ "title" ], ReadonlyTitlePostWithAbstractParent.readonly_attributes
```

```
  assert_nothing_raised do
```

```
    ReadonlyTitlePostWithAbstractParent.new(title: "can change this until you save")
```

```
  end
```

```
end
```



```
class ReadonlyTitleAbstractPost < ActiveRecord::Base
```

```
  self.abstract_class = true
```

```
  attr_readonly :title
```

```
end
```

```
class ReadonlyTitlePostWithAbstractParent < ReadonlyTitleAbstractPost
```

```
  self.table_name = "posts"
```

```
end
```

```
def test_readonly_attributes_in_abstract_class_descendant
```

```
  assert_equal [ "title" ], ReadonlyTitlePostWithAbstractParent.readonly_attributes
```

```
  assert_nothing_raised do
```

```
    ReadonlyTitlePostWithAbstractParent.new(title: "can change this until you save")
```

```
  end
```

```
end
```

```
$ git add . && git commit
```

Fix undefined attribute method with attr_readonly

The problem is where an abstract class defines a readonly attribute. When another class subclasses that parent with ``raise_on_assign_to_attr_readonly``, it will define the readonly method before the abstract parent has defined its attributes. As a result, when those methods are defined, they will not define the readonly attribute writer, so that ``super`` called from the readonly accessor will fail with a ``NoMethodError``.

Instead of trying to make the overridden attribute method lazier by defining it after the real attribute method has been defined, this commit changes the method override to be one layer down on `_write_attribute`. This avoids the issue of the attribute method being undefined because the real `_write_attribute` will always be defined.

Co-authored-by: Adrianna Chang <adrianna.chang@shopify.com>

Co-authored-by: Chris Salzberg <chris.salzberg@shopify.com>

Additional information

Closes [#46598](#)

Additional information

Closes [#46598](#)

Checklist

Before submitting the PR make sure the following are checked:

- * ☐ This Pull Request is related to one change.
- * ☐ Commit message has a detailed description of what changed and why.
- * ☐ Tests are added or updated if you fix a bug or add a feature.
- * ☐ CHANGELOG files are updated for the changed libraries.

Additional information

Closes [#46598](#)

Checklist

Before submitting the PR make sure the following are checked:

- * ☒ This Pull Request is related to one change.
- * ☒ Commit message has a detailed description of what changed and why.
- * ☒ Tests are added or updated if you fix a bug or add a feature.
- * ☐ CHANGELOG files are updated for the changed libraries.

Fix undefined attribute method with attr_readonly #46602

Merged

rafaelfranca merged 1 commit into rails:main from hmcguire-shopify:fix-lazy-readonly-writer on Nov 28, 2022



Conversation 0



Commits 1



Checks 3



Files changed 4



hmcguire-shopify commented on Nov 28, 2022 • edited

Contributor ...

Motivation / Background

The problem is where an abstract class defines a readonly attribute. When another class subclasses that parent with `raise_on_assign_to_attr_readonly`, it will define the readonly method before the abstract parent has defined its attributes. As a result, when those methods are defined, they will not define the readonly attribute writer, so that `super` called from the readonly accessor will fail with a `NoMethodError`.

Detail

Instead of trying to make the overridden attribute method lazier by defining it after the real attribute method has been defined, this commit changes the method override to be one layer down on `_write_attribute`. This avoids the issue of the attribute method being undefined because the real `_write_attribute` will always be defined.

Additional information

Closes #46598

Co-authored-by: Adrianna Chang adrianna.chang@shopify.com

Co-authored-by: Chris Salzberg chris.salzberg@shopify.com

<https://github.com/rails/rails/pull/46602>

`skipkayhil.github.io/slides/2023-railsconf.pdf`

Thank you!