

# Reflecting on Active Record Associations

Daniel Colson



@composerinteralia



@dodecadaniel



RailsConf  
Portland

```
class Repository < ApplicationRecord
  has_many :pull_requests
end
```

```
class PullRequest < ApplicationRecord
  belongs_to :repository
end
```

- repository
- repository=(repository)
- build\_repository
- create\_repository
- create\_repository!
- reload\_repository
- repository\_changed?
- repository\_previously\_changed?

- repository
- repository=(repository)
- build\_repository
- create\_repository
- create\_repository!
- reload\_repository
- repository\_changed?
- repository\_previously\_changed?
- Presence Validations
- Caching
- Autosaving
- Preloading
- Destroy Callbacks
- Scopes
- Extensions
- Etc.

# Rails Magic





**Let's Study Magic!**

# Daniel Colson

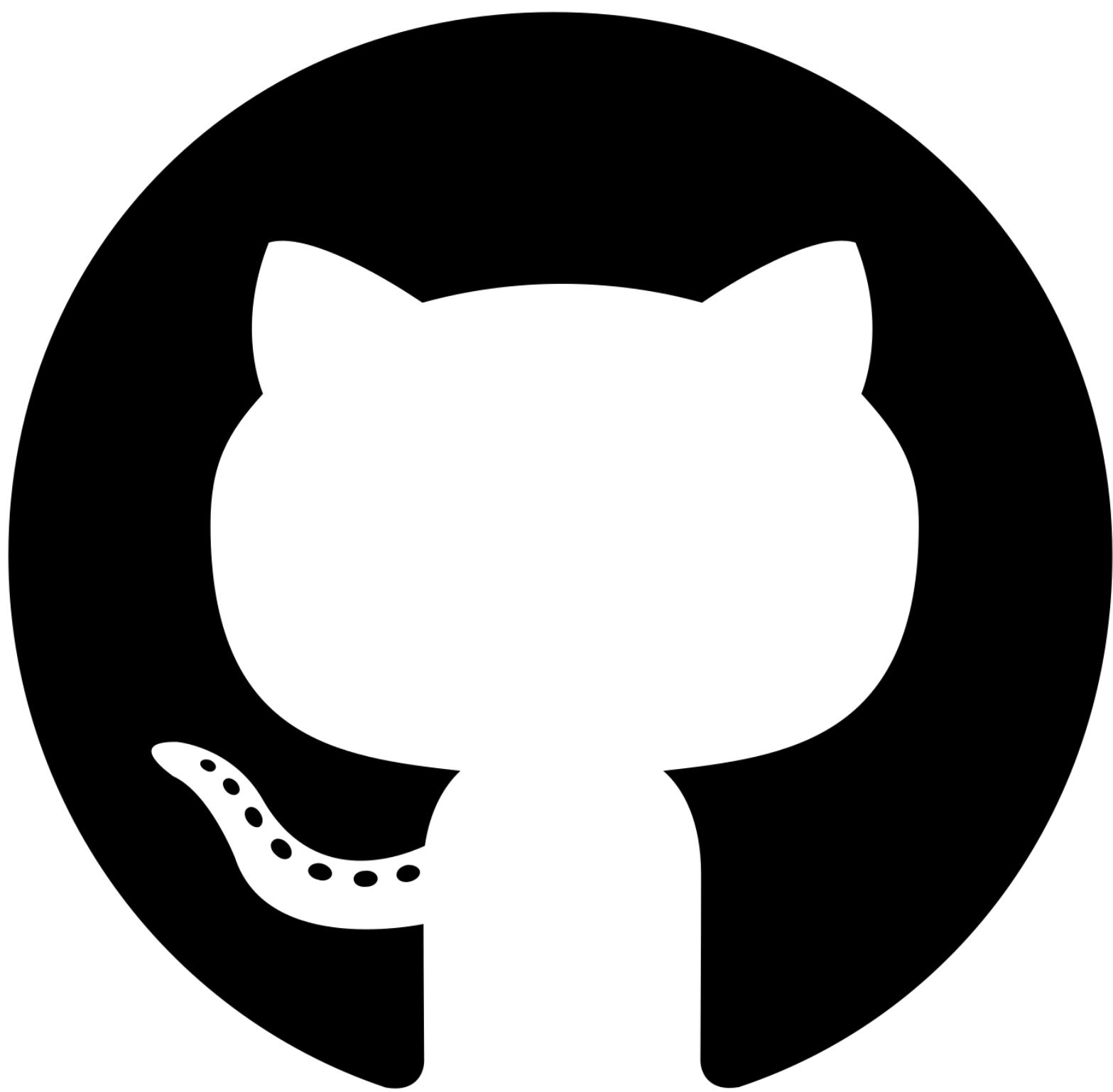


@composerinteralia



@dodecadaniel





# Study Guide

---

- Metaprogramming
- Reflections
- Caching
- Relations
- Inverses

# Metaprogramming



# Metaprogramming

---

```
class PullRequest < ApplicationRecord
  belongs_to :repository
end
```

# Metaprogramming

---

```
class PullRequest < ApplicationRecord  
  self.belongs_to(:repository)  
end
```



Calling a class method

# Metaprogramming

---

```
class PullRequest < ApplicationRecord
  def self.belongs_to(name) ← Defining a class method
end
end
```

# Metaprogramming

---

## Reader Method

# Metaprogramming

---

## Reader Method

```
> pull_request  
=> <#PullRequest @repository_id=42>
```

# Metaprogramming

---

## Reader Method

```
> pull_request  
=> <#PullRequest @repository_id=42>
```

```
> pull_request.repository  
=> <#Repository @id=42>
```

# Metaprogramming

---

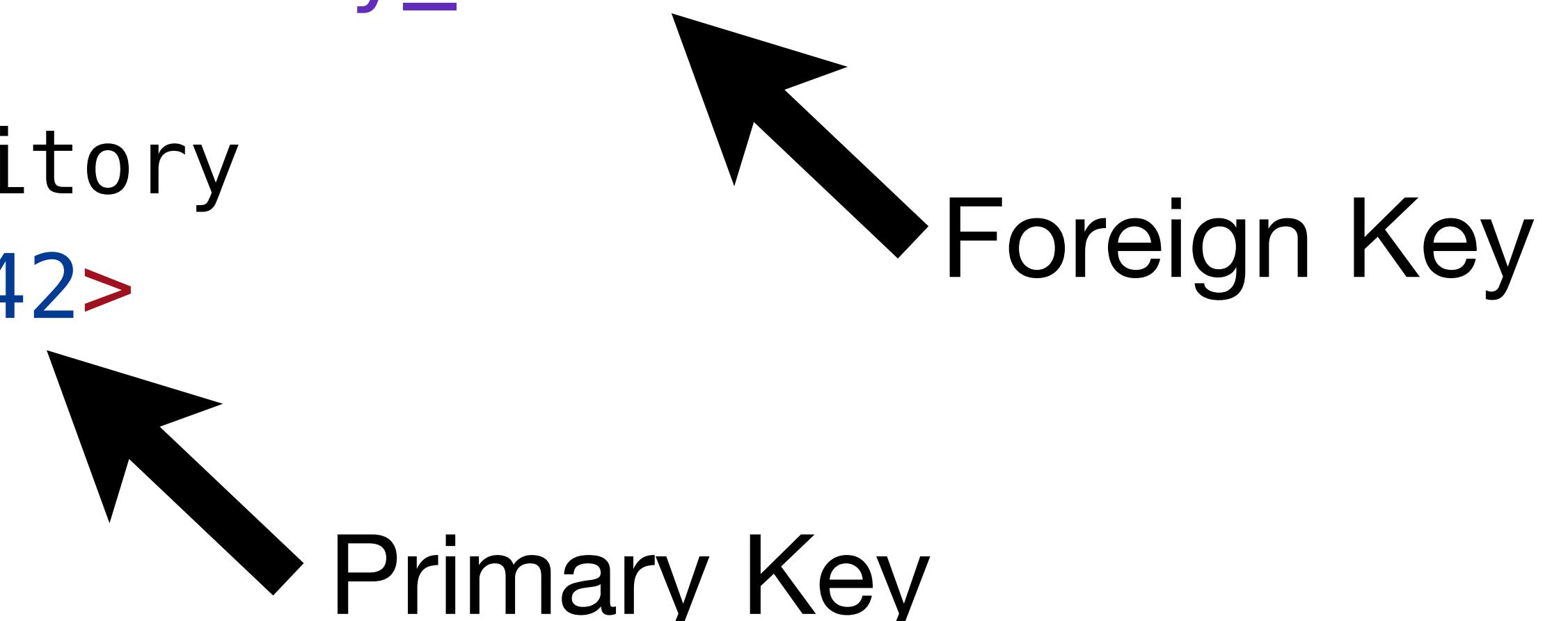
## Reader Method

```
> pull_request
```

```
=> <#PullRequest @repository_id=42>
```

```
> pull_request.repository
```

```
=> <#Repository @id=42>
```



# Metaprogramming

---

## Reader Method

```
def self.belongs_to(name)
```

```
  def repository
```

```
    Repository.where(id: repository_id).first
```

```
  end
```

```
end
```



# Metaprogramming

---

Writer Method

# Metaprogramming

---

## Writer Method

```
> pull_request  
=> <#PullRequest @repository_id=42>
```

# Metaprogramming

---

## Writer Method

> pull\_request

=> <**#PullRequest** @repository\_id=42>

> repository

=> <**#Repository** @id=77>

# Metaprogramming

---

## Writer Method

```
> pull_request
```

```
=> <#PullRequest @repository_id=42>
```

```
> repository
```

```
=> <#Repository @id=77>
```

```
> pull_request.repository = repository
```

```
=> <#PullRequest @repository_id=77>
```

# Metaprogramming

---

## Writer Method

```
def self.belongs_to(name)
  def repository
    Repository.where(id: repository_id).first
  end
```

```
def repository=(repository)
  self.repository_id = repository.id
end
```



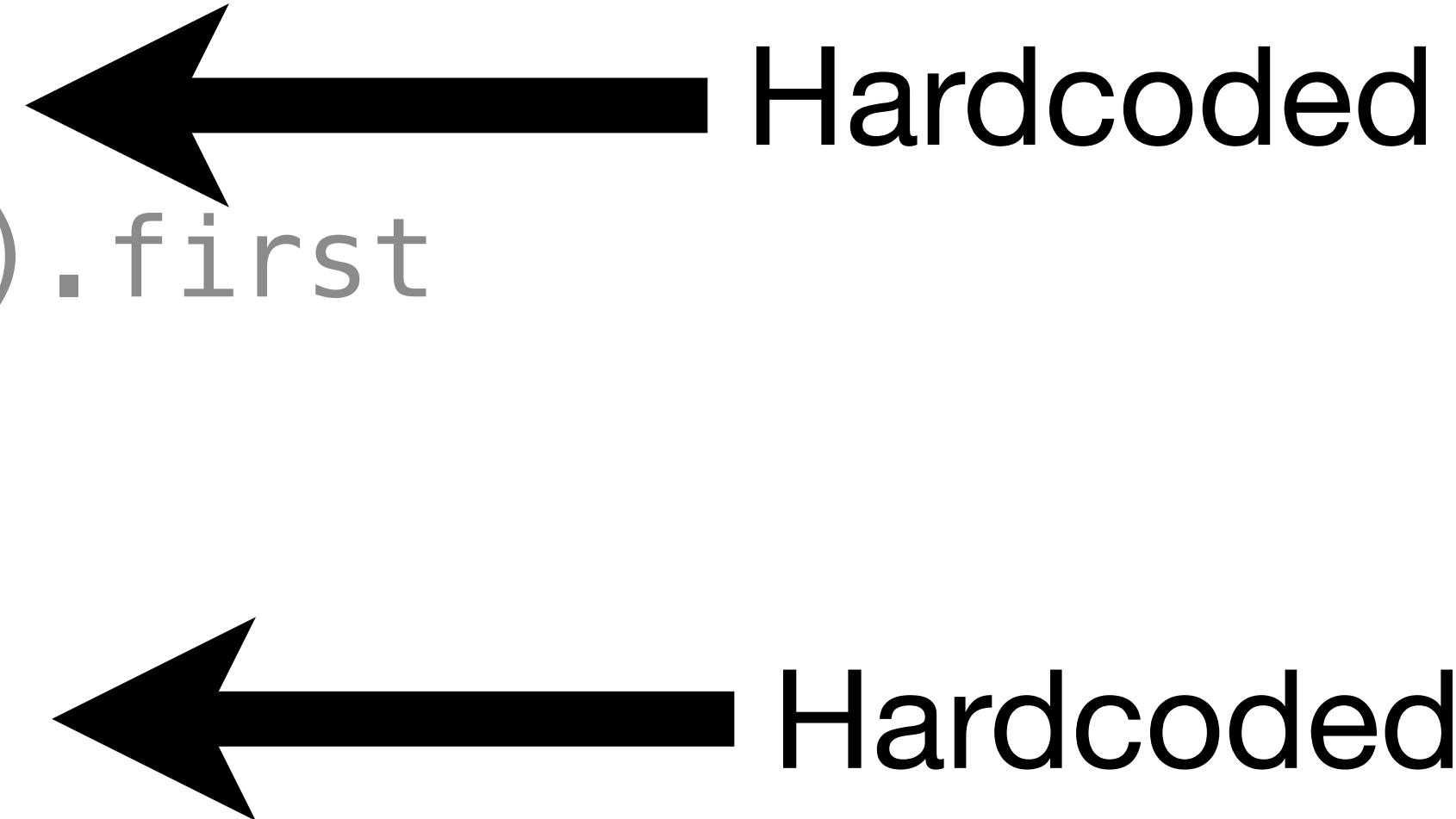
```
end
```

# Metaprogramming

---

```
def self.belongs_to(name)
  def repository
    Repository.where(id: repository_id).first
  end

  def repository=(record)
    self.repository_id = repository.id
  end
end
```



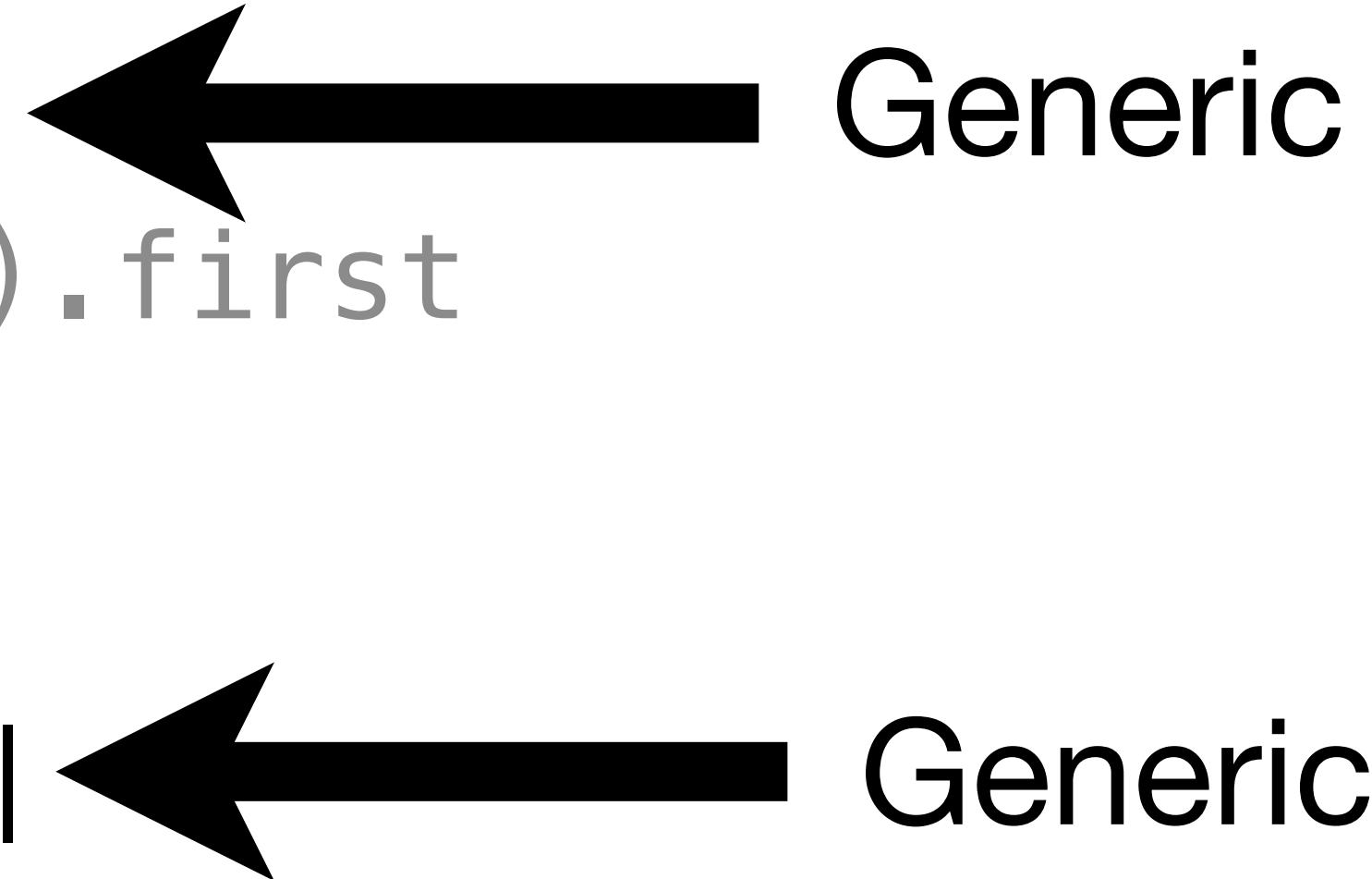
The diagram illustrates two instances of hardcoding. Two black arrows point from the word "repository" in the first code block to the text "Hardcoded" in the second code block, indicating that the variable name "repository" is being used in a hardcoded manner.

# Metaprogramming

---

```
def self.belongs_to(name)
  define_method(name) do
    Repository.where(id: repository_id).first
  end

  define_method("#{name}=") do |record|
    self.repository_id = repository.id
  end
end
```



Generic

Generic

# Metaprogramming

---

```
def self.belongs_to(name)
  define_method(name) do
    Repository.where(:id => self[:repository_id]).first
  end
```

Hardcoded

```
define_method("#{name}=") do |record|
  self[:repository_id] = record[:id]
end
end
```

# Metaprogramming

---

```
def self.belongs_to(name)
  define_method(name) do
    klass.where(primary_key => self[foreign_key]).first
  end

  define_method("#{name}=") do |record|
    self[foreign_key] = record[primary_key]
  end
end
```

Generic

# Metaprogramming

---

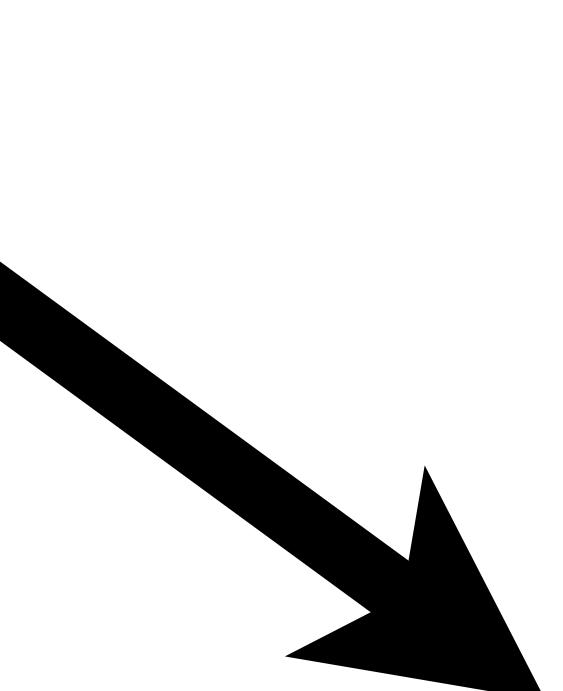
```
class PullRequest < ApplicationRecord
  belongs_to :repository
end
```

# Metaprogramming

---

```
class PullRequest < ApplicationRecord  
  belongs_to :repository  
end
```

?



```
klass          => Repository  
primary_key  => :id  
foreign_key  => :repository_id
```

# Reflections



# Reflections

---

```
class Reflection
  def initialize(name, active_record)
    @active_record = active_record
    @name          = name
  end
end
```

# Reflections

---

## Reflection

---

```
@active_record = PullRequest  
@name           = :repository
```

# Reflections

---

## Reflection

---

```
@active_record = PullRequest  
@name           = :repository
```

- `#klass`
- `#primary_key`
- `#foreign_key`

# Reflections

---

## Reflection

```
@active_record = PullRequest  
@name           = :repository
```

- `#klass => Repository` ←
- `#primary_key`
- `#foreign_key`

# Reflections

---

## Reflection

```
@active_record = PullRequest  
@name           = :repository
```

- `#klass => Repository` ←
- `#primary_key`
- `#foreign_key`

# Reflections

---

```
class Reflection
  def klass
    @name.to_s.camelize.constantize
  end
end
```



# Reflections

---

## Reflection

```
@active_record = PullRequest  
@name           = :repository
```

- #klass => Repository
- #primary\_key => :id
- #foreign\_key



# Reflections

---

```
class Reflection
  def primary_key
    klass.primary_key ←
  end
end
```

# Reflections

---

## Reflection

```
@active_record = PullRequest  
@name           = :repository
```

- #klass => Repository
- #primary\_key => :id
- #foreign\_key => :repository\_id



# Reflections

---

## Reflection

```
@active_record = PullRequest  
@name           = :repository
```

- #klass => Repository
- #primary\_key => :id
- #foreign\_key => :repository\_id



# Reflections

---

```
class Reflection
  def foreign_key
    "#{@name}_id" ←
  end
end
```

# Reflections

---

```
def self.belongs_to(name)
  reflection = Reflection.new(name, self)
```

# Reflections

---

```
def self.belongs_to(name)
  reflection = Reflection.new(name, self)
  klass      = reflection.klass
  primary_key = reflection.primary_key
  foreign_key = reflection.foreign_key
```

# Reflections

---

```
def self.belongs_to(name)
  define_method(name) do
    klass.where(primary_key => self[foreign_key]).first
  end

  define_method("#{name}=") do |record|
    self[foreign_key] = record[primary_key]
  end
end
```

Caching 💰

# Caching

---

```
> repo = pull_request.repository
```

```
=> #<Repository:0x307468 @id=42>
```

```
> same_repo = pull_request.repository
```

```
=> #<Repository:0x647570 @id=42>
```

# Caching

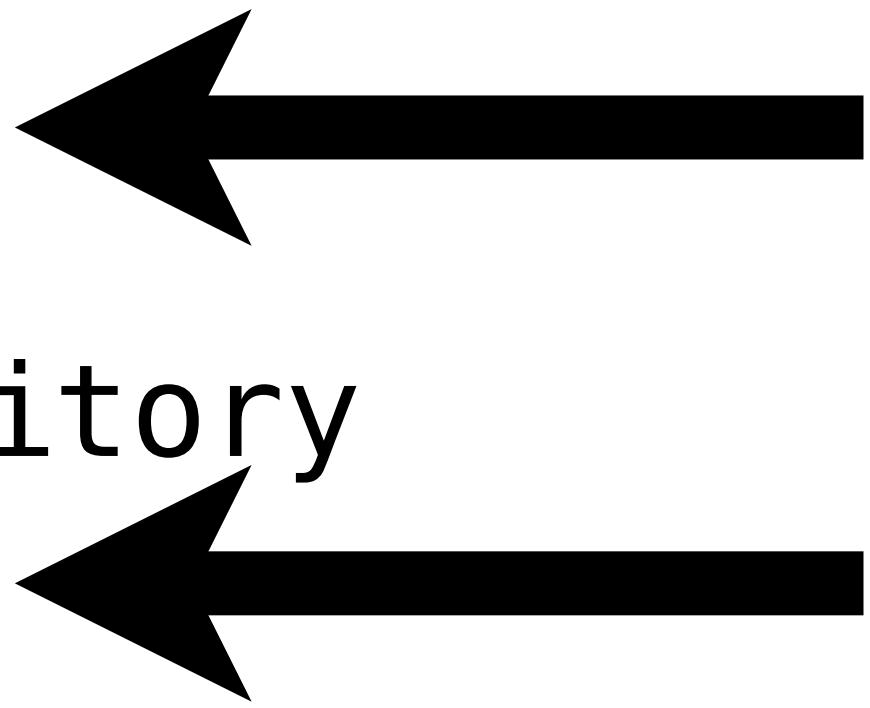
---

```
> repo = pull_request.repository
```

```
=> #<Repository:0x307468 @id=42>
```

```
> same_repo = pull_request.repository
```

```
=> #<Repository:0x647570 @id=42>
```



Different objects

# Caching

---

```
> repo = pull_request.repository
```

```
=> #<Repository:0x307468 @id=42>
```

```
> same_repo = pull_request.repository
```

```
=> #<Repository:0x647570 @id=42>
```

```
> repo.name = "new_name"
```

```
> same_repo.name
```

```
=> "old_name"
```



Different objects

# Caching

---

```
> repo = pull_request.repository
```

```
=> #<Repository:0x307468 @id=42>
```

```
> same_repo = pull_request.repository
```

```
=> #<Repository:0x647570 @id=42>
```

```
> repo.name = "new_name"
```

```
> same_repo.name
```

```
=> "old_name"
```



Different objects

Inconsistent data



# Caching

---

```
class Association
  def initialize(owner, reflection)
    @owner          = owner
    @reflection     = reflection
    @loaded         = false
    @target         = nil
  end
end
```

# Caching

---

```
def self.belongs_to(name)
  define_method(name) do
    klass.where(primary_key => self[foreign_key]).first
  end

  define_method("#{name}=") do |record|
    self[foreign_key] = record[primary_key]
  end
end
```

# Caching

---

```
def self.belongs_to(name)
  define_method(name) do
    [REDACTED]
  end
```

```
  define_method("#{name}=") do |record|
    [REDACTED]
  end
end
```

# Caching

---

```
class Association
  def reader
    
  end

  def writer(record)
    
  end
end
```

The diagram illustrates the implementation of a caching mechanism. It shows two methods, `reader` and `writer(record)`, defined within the `Association` class. Each method is preceded by a large black arrow pointing left, labeled **Association Reader** above the first one and **Association Writer** above the second one. Below each arrow is a rectangular callout box with a black border, which encloses the body of the respective method definition.

# Caching

---

```
class Association
  def reader
    klass.where(primary_key => @owner[foreign_key]).first
  end

  def writer(record)
    @owner[foreign_key] = record[primary_key]
  end
end
```

Association Reader

Association Writer

# Caching

---

```
def self.belongs_to(name)
  define_method(name) do
    association(name).reader
  end

  define_method("#{name}=") do |record|
    association(name).writer(record)
  end
end
```



Use Association  
Reader and Writer



# Caching

---

```
def self.belongs_to(name)
  define_method(name) do
    association(name).reader
  end

  define_method("#{name}=") do |record|
    association(name).writer(record)
  end
end
```



Use Association  
Reader and Writer



# Caching

---

## Association

---

`@owner` = #<PullRequest>

`@reflection` = #<Reflection>

`@loaded` = false

`@target` = nil

# Caching

---

```
def reader
  if loaded?
    else
      end
    end
```

## Association

---

@owner	= #<PullRequest>
@reflection	= #<Reflection>
@loaded	= false
@target	= nil

# Caching

---

```
def reader
  if loaded? ←
    else
      end
    end
```

## Association

---

@owner	= #<PullRequest>
@reflection	= #<Reflection>
@loaded	= false
@target	= nil

# Caching

---

```
def reader
  if loaded?
    else
      self.target = klass.where(...).first
    end
  end
```

## Association

---

@owner	= #<PullRequest>
@reflection	= #<Reflection>
@loaded	= false
@target	= nil

# Caching

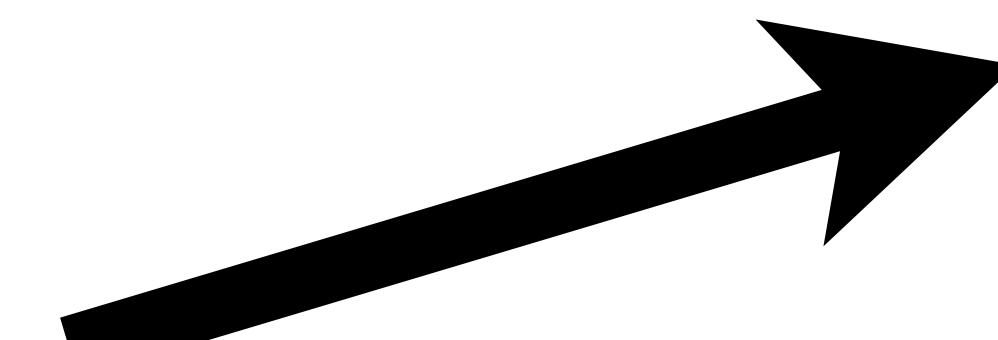
---

```
def reader
  if loaded?
    ...
  else
    self.target = klass.where(...).first
  end
end
```

## Association

---

@owner	= #<PullRequest>
@reflection	= #<Reflection>
@loaded	= true
@target	= #<Repository>



# Caching

---

```
def reader
  if loaded? ←
    ...
  else
    self.target = klass.where(...).first
  end
end
```

## Association

---

@owner	= #<PullRequest>
@reflection	= #<Reflection>
@loaded	= true
@target	= #<Repository>

# Caching

---

```
def reader
  if loaded?
    target ←
  else
    self.target = klass.where(...).first
  end
end
```

## Association

---

@owner	= #<PullRequest>
@reflection	= #<Reflection>
@loaded	= true
@target	= #<Repository>

# Caching

---

Beware of a stale cache!

```
def writer(record)
  @owner[foreign_key] = record[primary_key]
  self.target = record ← Update target
end
```

# Caching

---

```
> repo = pull_request.repository
```

```
=> #<Repository:0x776f6e @id=42>
```

```
> same_repo = pull_request.repository
```

```
=> #<Repository:0x776f6e @id=42>
```

# Caching

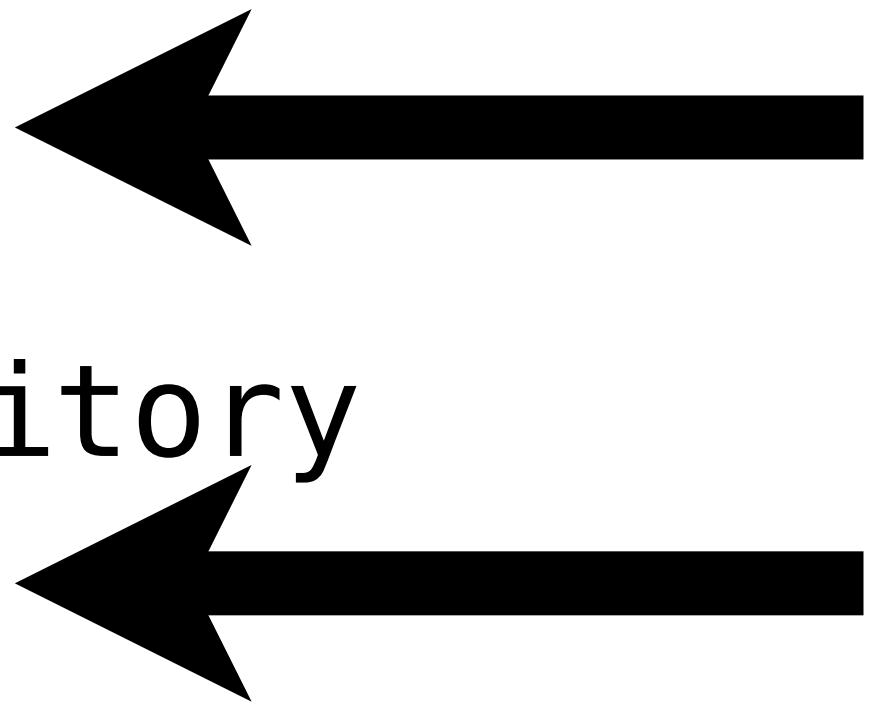
---

```
> repo = pull_request.repository
```

```
=> #<Repository:0x776f6e @id=42>
```

```
> same_repo = pull_request.repository
```

```
=> #<Repository:0x776f6e @id=42>
```



Same object

# Caching

---

```
> repo = pull_request.repository
```

```
=> #<Repository:0x776f6e @id=42>
```

```
> same_repo = pull_request.repository
```

```
=> #<Repository:0x776f6e @id=42>
```

```
> repo.name = "new_name"
```

```
> same_repo.name
```

```
=> "new_name"
```



Same object

# Caching

---

```
> repo = pull_request.repository
```

```
=> #<Repository:0x776f6e @id=42>
```

```
> same_repo = pull_request.repository
```

```
=> #<Repository:0x776f6e @id=42>
```

```
> repo.name = "new_name"
```

```
> same_repo.name
```

```
=> "new_name"
```



Consistent data 

Same object

# Relations



# Relations

---

```
class Repository < ApplicationRecord
  has_many :pull_requests
end
```

# Relations

---

```
def self.has_many(name)
  define_method(name) do
    association(name).reader
  end

  define_method("#{name}=") do |record|
    association(name).writer(record)
  end
end
```

# Relations

---

```
class BelongsToAssociation < Association  
end
```

```
class HasManyAssociation < Association  
end
```

# Relations

---

## BelongsToAssociation#reader

```
def reader
  if loaded?
    target
  else
    self.target =
      klass.where(primary_key => @owner[foreign_key]).first
  end
end
```

# Relations

---

## BelongsToAssociation#reader

```
def reader
  if loaded?
    target
  else
    self.target =
      klass.where(primary_key => @owner[foreign_key]).first
  end
end
```

The diagram illustrates the relationship between the primary key and foreign key in the database. Two large black arrows point from the text "Primary Key" and "Foreign Key" to the corresponding parameters in the highlighted code: "primary\_key" and "@owner[foreign\_key]" respectively.

# Relations

---

## BelongsToAssociation#reader

```
def reader
  if loaded?
    target
  else
    self.target =
      klass.where(primary_key => @owner[foreign_key]).first
  end
end
```

Single Record

Primary Key

Foreign Key

# Relations

---

## HasManyAssociation#reader

```
def reader
  if loaded?
    target
  else
    self.target =
      klass.where(foreign_key => @owner[primary_key]).to_a
  end
end
```

# Relations

---

## HasManyAssociation#reader

```
def reader
  if loaded?
    target
  else
    self.target =
      klass.where(foreign_key => @owner[primary_key]).to_a
  end
end
```

Foreign Key      Primary Key

# Relations

---

## HasManyAssociation#reader

```
def reader
  if loaded?
    target
  else
    self.target =
      klass.where(foreign_key => @owner[primary_key]).to_a
  end
end
```

Array of Records

Foreign Key

Primary Key

The diagram illustrates the execution flow of the Ruby code. A large horizontal arrow points from the boxed query code to the 'Array of Records' label at the top right. Two smaller arrows point upwards from the 'Foreign Key' and 'Primary Key' labels to their respective parts in the query code: 'foreign\_key' and '@owner[primary\_key]'. This visualizes how the database query is mapped into an array of records.

# Relations

---

```
> repository.pull_requests  
=> SELECT * FROM pull_requests WHERE repository_id = 42  
=> [#<PullRequest @repository_id=42>]
```

# Relations

---

```
> repository.pull_requests
=> SELECT * FROM pull_requests WHERE repository_id = 42
=> [#<PullRequest @repository_id=42>]

> repository.pull_requests
=> [#<PullRequest @repository_id=42>]
```

# Relations

---

What is a Relation?

# Relations

---

What is a Relation?

```
> pull_requests = PullRequest.where(repository_id: 42)
=> <#ActiveRecord::Relation>
```

# Relations

---

## What is a Relation?

```
> pull_requests = PullRequest.where(repository_id: 42)
=> <#ActiveRecord::Relation>
```

- Super-powered array of records
- Lazy loading
- #create
- And much more

# Relations

---

## Association Relations?

```
> pull_requests = repository.pull_requests  
=> <#ActiveRecord::Relation>
```

# Relations

---

## Association Relations?

```
> pull_requests = repository.pull_requests  
=> <#ActiveRecord::Relation>
```

### Association

---

@loaded	= false
@target	= nil

# Relations

---

## Association Relations?

```
> pull_requests = repository.pull_requests
```

```
=> <#ActiveRecord::Relation>
```

```
> pull_requests.to_a
```

### Association

---

@loaded = false

@target = nil

# Relations

---

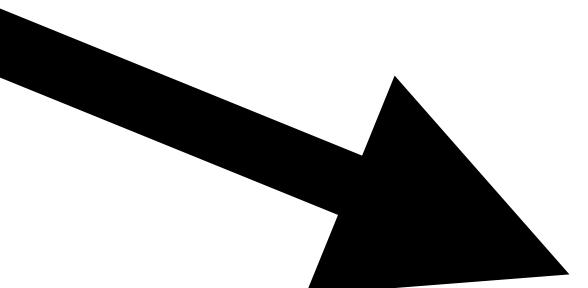
## Association Relations?

```
> pull_requests = repository.pull_requests
```

```
=> <#ActiveRecord::Relation>
```

```
> pull_requests.to_a
```

```
=> [<#PullRequest>]
```



### Association

---

@loaded = true

@target = [<#PullRequest>]

# Relations

---

```
class CollectionProxy < ActiveRecord::Relation
  def initialize(association)
    @association = association
    super
  end
end
```

# Relations

---

HasManyAssociation#reader

```
def reader  
  CollectionProxy.new(self)  
end
```

Updated Reader



# Relations

---

## HasManyAssociation#load\_target

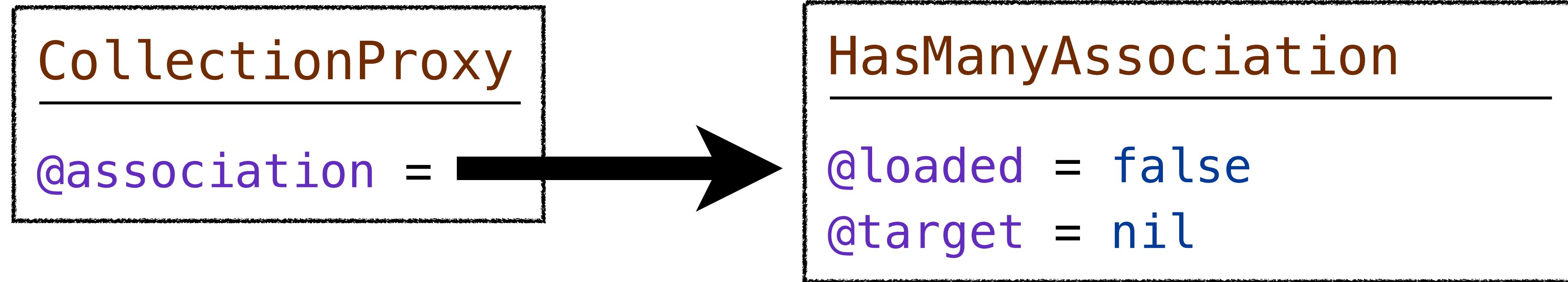
```
def load_target
```

```
  if loaded?
    target
  else
    self.target =
      klass.where(foreign_key => @owner[primary_key]).to_a
  end
end
```

Moved From Reader

# Relations

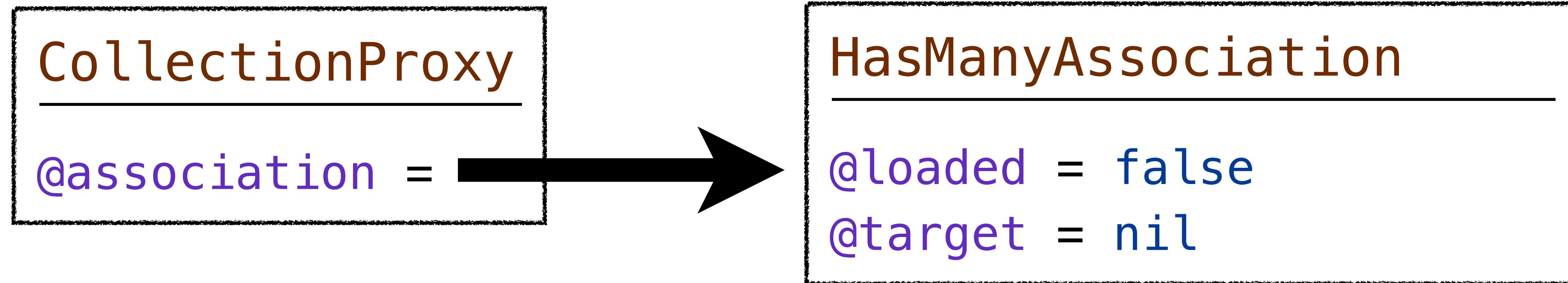
---



`repository.pull_requests`

# Relations

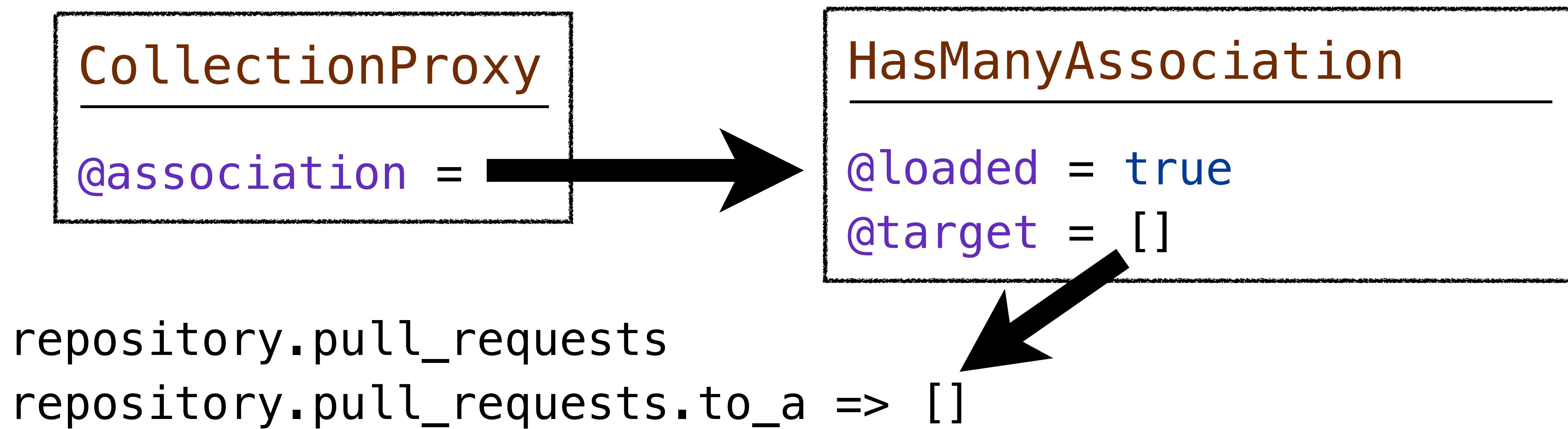
---



`repository.pull_requests`  
`repository.pull_requests.to_a`

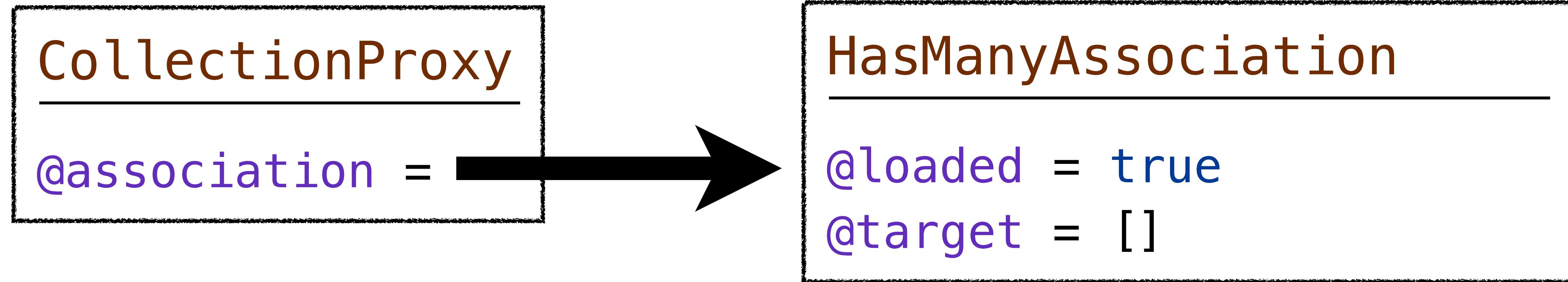
# Relations

---



# Relations

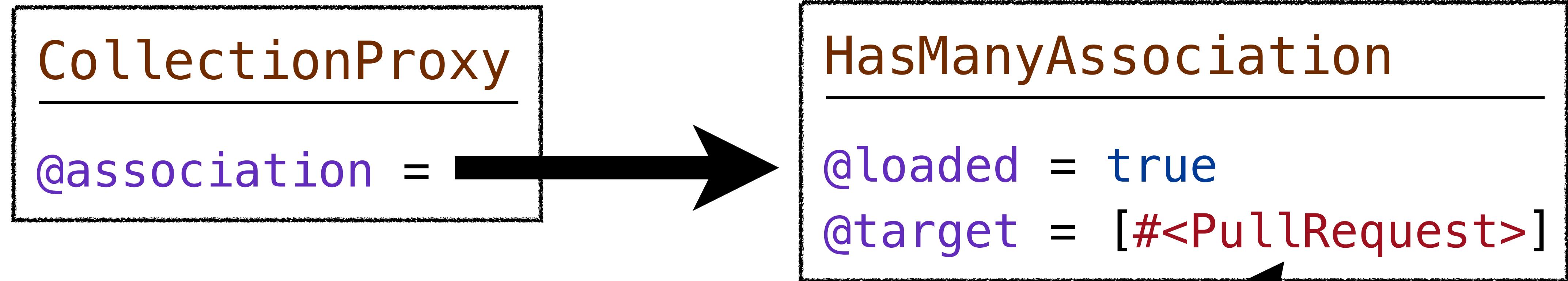
---



```
repository.pull_requests
repository.pull_requests.to_a => []
repository.pull_requests.create
```

# Relations

---



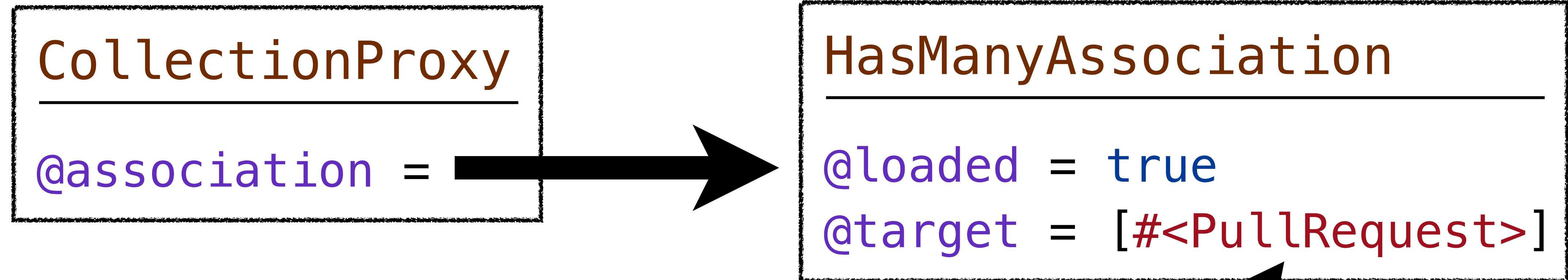
`repository.pull_requests`

`repository.pull_requests.to_a => []`

`repository.pull_requests.create => #<PullRequest>`

# Relations

---



```
repository.pull_requests
repository.pull_requests.to_a => []
repository.pull_requests.create => #<PullRequest>
repository.pull_requests.to_a => [#<PullRequest>]
```

Inverses



# Inverses

---

```
class Repository < ApplicationRecord
  has_many :pull_requests
end
```

```
class PullRequest < ApplicationRecord
  belongs_to :repository
end
```

# Inverses

---

```
> pull_requests = repository.pull_requests  
=> SELECT * FROM pull_requests WHERE repository_id = 42
```

# Inverses

---

```
> pull_requests = repository.pull_requests
=> SELECT * FROM pull_requests WHERE repository_id = 42
> pull_requests.map(&:repository)
```

# Inverses

---

```
> pull_requests = repository.pull_requests
=> SELECT * FROM pull_requests WHERE repository_id = 42

> pull_requests.map(&repository)
=> SELECT * FROM repositories WHERE id = 42 LIMIT 1
=> SELECT * FROM repositories WHERE id = 42 LIMIT 1
=> SELECT * FROM repositories WHERE id = 42 LIMIT 1
=> SELECT * FROM repositories WHERE id = 42 LIMIT 1
=> SELECT * FROM repositories WHERE id = 42 LIMIT 1
```



# Inverses

---

## BelongsToAssociation

@owner	= #<PullRequest>
@loaded	= false
@target	= nil

# Inverses

---

`pull_request.repository`

## `BelongsToAssociation`

---

<code>@owner</code>	= <code>#&lt;PullRequest&gt;</code>
<code>@loaded</code>	= <code>false</code>
<code>@target</code>	= <code>nil</code>

# Inverses

---

`pull_request.repository`

## BelongsToAssociation

<code>@owner</code>	= <code>#&lt;PullRequest&gt;</code>
<code>@loaded</code>	= <code>false</code>
<code>@target</code>	= <code>nil</code>

## hasManyAssociation

<code>@owner</code>	= <code>#&lt;Repository&gt;</code>
<code>@loaded</code>	= <code>true</code>
<code>@target</code>	= <code>[#&lt;PullRequest&gt;, ...]</code>

`repository.pull_requests`

# Inverses

---

`pull_request.repository`

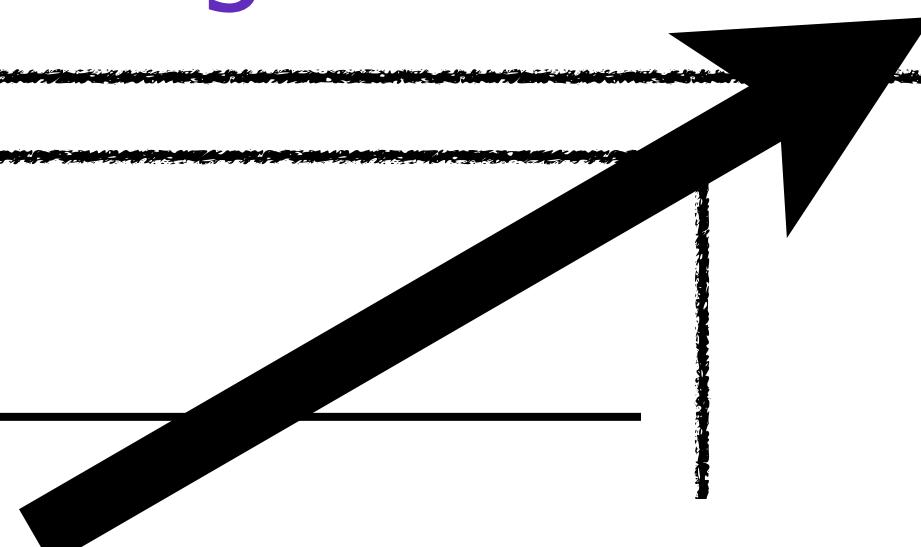
## BelongsToAssociation

<code>@owner</code>	= #<PullRequest>
<code>@loaded</code>	= true
<code>@target</code>	= #<Repository>

## hasManyAssociation

<code>@owner</code>	= #<Repository>
<code>@loaded</code>	= true
<code>@target</code>	= [#<PullRequest>, ...]

`repository.pull_requests`



# Inverses

---

## HasManyAssociation

---

```
@owner      = #<Repository>
@loaded     = true
@target     = [#<PullRequest>, ...]
```

# Inverses

---

```
@target.each do |record|
```

## HasManyAssociation

---

```
@owner      = #<Repository>
@loaded     = true
@target     = [#<PullRequest>, ...]
```

# Inverses

---

```
@target.each do |record| ← #<PullRequest>
```

## HasManyAssociation

---

```
@owner      = #<Repository>
@loaded     = true
@target     = [#<PullRequest>, ...]
```

# Inverses

---

```
@target.each do |record|  
  association = record.association(:repository)
```

## HasManyAssociation

---

```
@owner      = #<Repository>  
@loaded     = true  
@target     = [#<PullRequest>, ...]
```

# Inverses

---

```
@target.each do |record| ← #<PullRequest>  
  association = record.association(:repository)  
  association.target = @owner
```

## HasManyAssociation

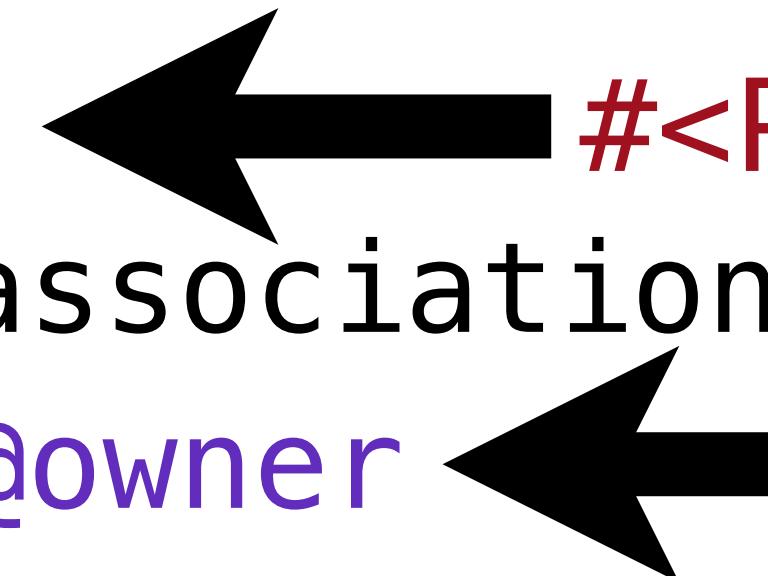
---

```
@owner      = #<Repository>  
@loaded      = true  
@target      = [#<PullRequest>, ...]
```

# Inverses

---

```
@target.each do |record|  
  association = record.association(:repository)  
  association.target = @owner
```



## HasManyAssociation

---

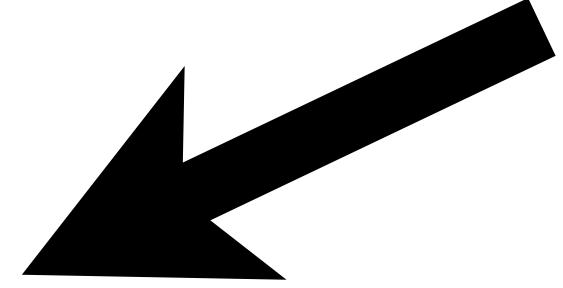
```
@owner      = #<Repository>  
@loaded     = true  
@target     = [#<PullRequest>, ...]
```

# Inverses

---

```
@target.each do |record|
  association = record.association(:repository)
  association.target = @owner
end
```

Harcoded



# Inverses

---

## Reflection

---

```
@active_record = Repository  
@name           = :pull_requests
```

# Inverses

---

## Reflection

---

```
@active_record = Repository  
@name           = :pull_requests
```

```
#inverse_name => :repository
```

# Inverses

---

## Reflection

```
@active_record = Repository  
@name           = :pull_requests
```

```
#inverse_name => :repository
```

# Inverses

---

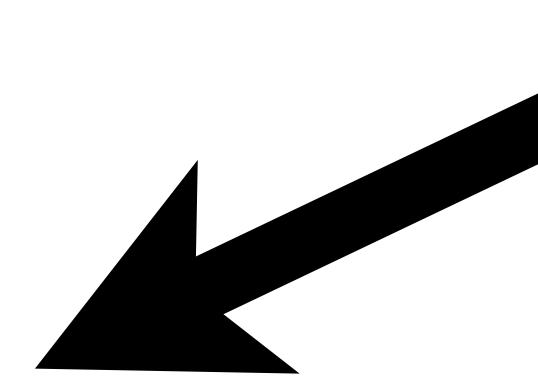
```
class Reflection
  def inverse_name
    @active_record.name.underscore ←
  end
end
```

# Inverses

---

```
@target.each do |record|
  association = record.association(:repository)
  association.target = @owner
end
```

Harcoded

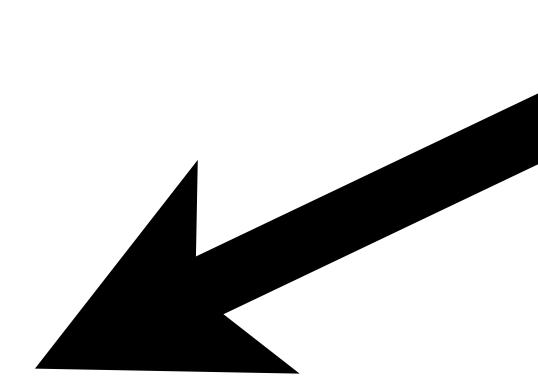


# Inverses

---

```
@target.each do |record|
  association = record.association(@reflection.inverse_name)
  association.target = @owner
end
```

Generic



# Inverses

---

```
> pull_requests = repository.pull_requests  
=> SELECT * FROM pull_requests WHERE repository_id = 42
```

# Inverses

---

```
> pull_requests = repository.pull_requests
=> SELECT * FROM pull_requests WHERE repository_id = 42
> pull_requests.map(&:repository)
```

# Inverses

---

```
> pull_requests = repository.pull_requests
=> SELECT * FROM pull_requests WHERE repository_id = 42
> pull_requests.map(&:repository)
```





# Further Studies

---

- Through Associations, Polymorphic Associations, Scoping, Etc.
- [github.com/rails/rails](https://github.com/rails/rails)
  - activerecord/lib/active\_record/associations.rb
  - activerecord/lib/active\_record/reflection.rb
  - activerecord/lib/active\_record/associations/\*
- [danieljamescolson.com/blog](http://danieljamescolson.com/blog)

# Practical Applications

---

- Use Rails features
- Write less custom code
- Write more efficient code
- Avoid inconsistent data
- Code with confidence

# Reflecting on Active Record Associations

Daniel Colson



@composerinteralia



@dodecadaniel



RailsConf  
Portland