RailsConf
Portland

Reflecting on Active Record Associations

Daniel Colson

@composerinteralia

@dodecadaniel

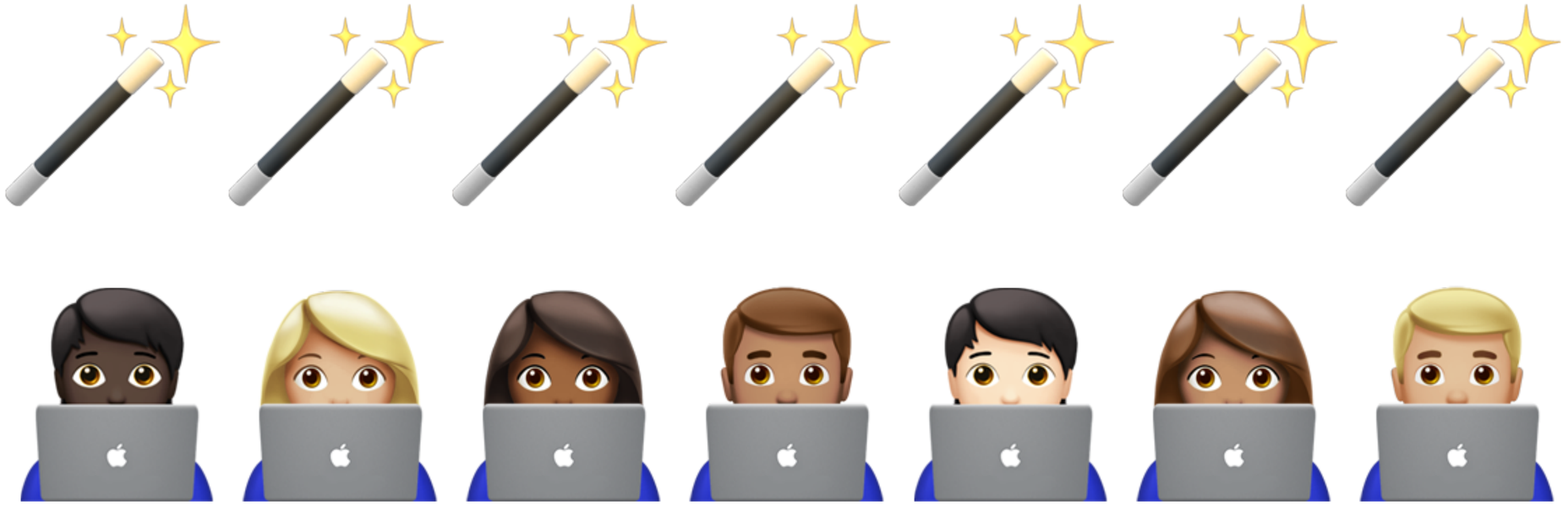
```ruby
class Repository < ApplicationRecord
  has_many :pull_requests
end

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 🪄

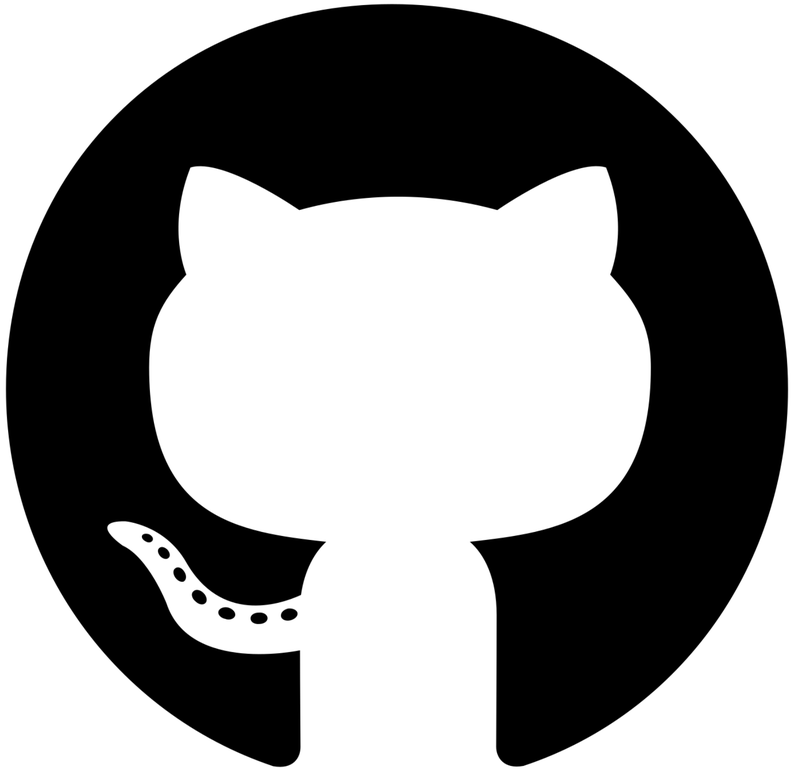


Let’s Study Magic!

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)
  end
end
```

Defining a class method

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
```

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

Hardcoded

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

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

Metaprogramming

Generic

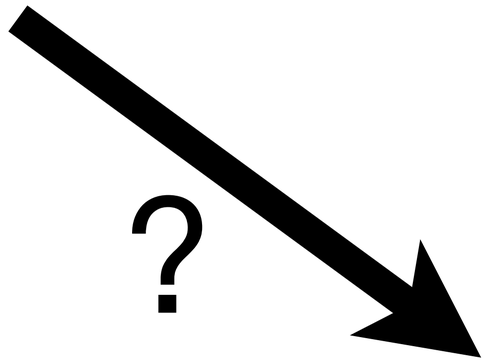
```
def self.belongs_to(name)
  define_method(name) do
    klass.where(primary_key => self[foreign_key]).first
  end

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

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

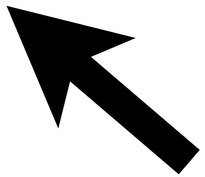
```
> repo = pull_request.repository
=> #<Repository:0x307468 @id=42>

> same_repo = pull_request.repository
=> #<Repository:0x647570 @id=42>

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

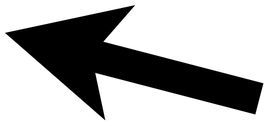
Different objects

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
    
  end

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

Caching

```
class Association
  def reader
    
  end

  def writer(record)
    
  end
end
```

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

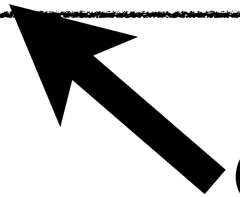
```
def self.belongs_to(name)
  define_method(name) do
    association(name).reader
  end

  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



Get Association by name

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
```

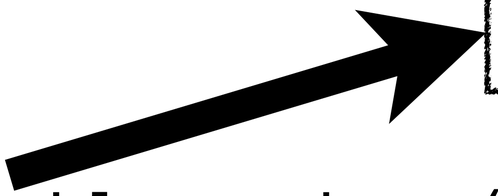
Caching

```
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
end
```

Update target

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

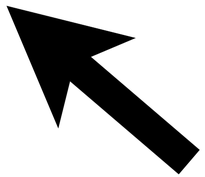
```
> repo = pull_request.repository
=> #<Repository:0x776f6e @id=42>

> same_repo = pull_request.repository
=> #<Repository:0x776f6e @id=42>

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

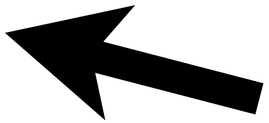
Same object

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"
```

Same object





Consistent data 👍🏻

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

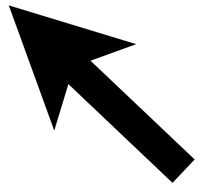
BelongsToAssociation#reader

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

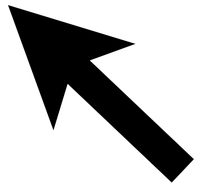
Relations

BelongsToAssociation#reader

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



Primary Key

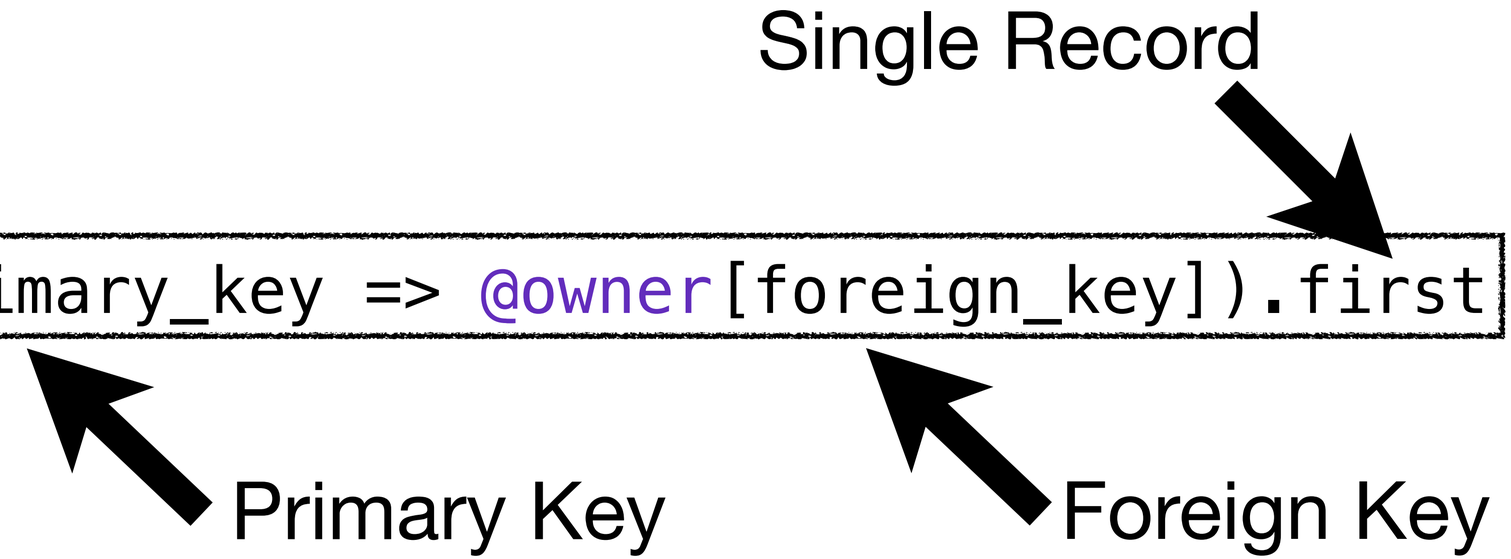


Foreign Key

Relations

BelongsToAssociation#reader

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



Relations

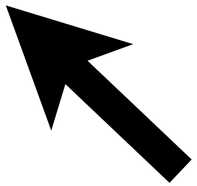
HasManyAssociation#reader

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

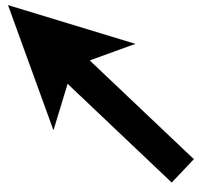
Relations

HasManyAssociation#reader

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



Foreign Key

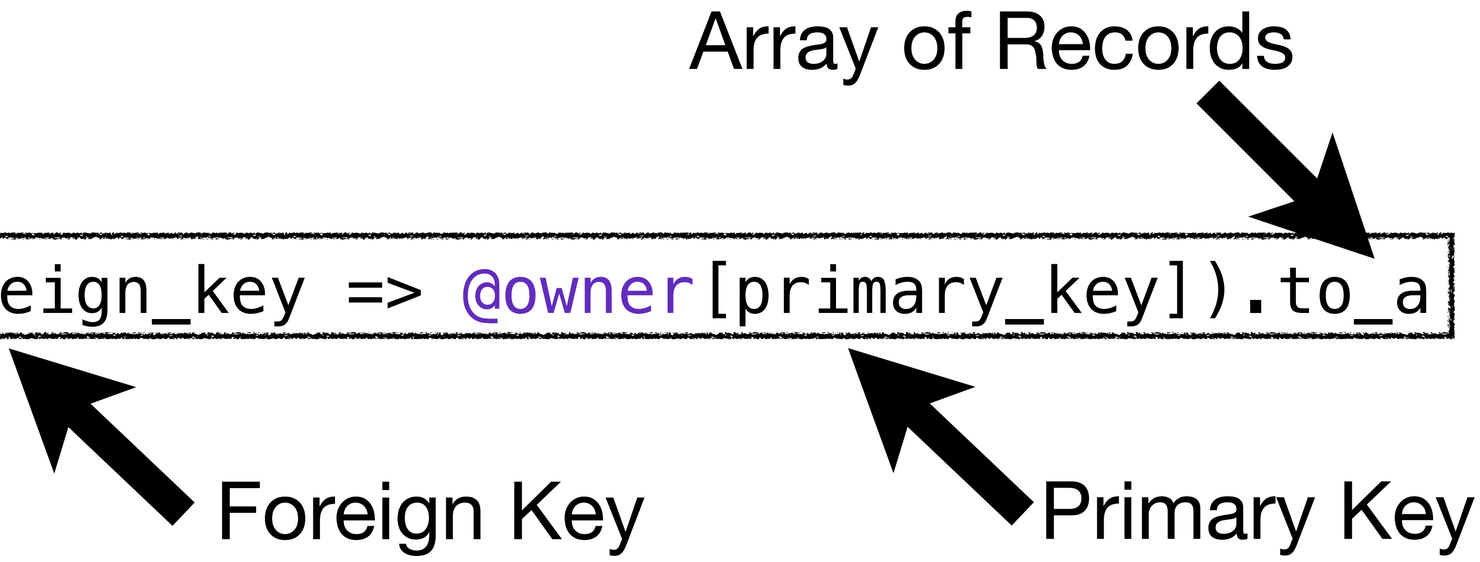


Primary Key

Relations

HasManyAssociation#reader

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



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?

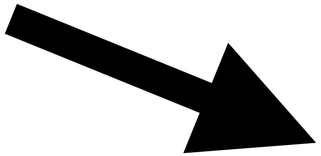
```
> pull_requests = repository.pull_requests
=> <#ActiveRecord::Relation>

> 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
```

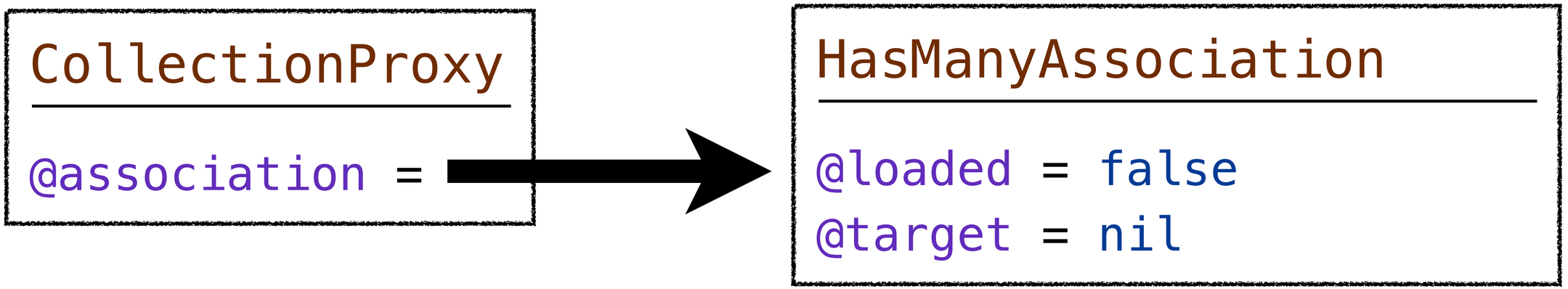
Relations

HasManyAssociation#load_target

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

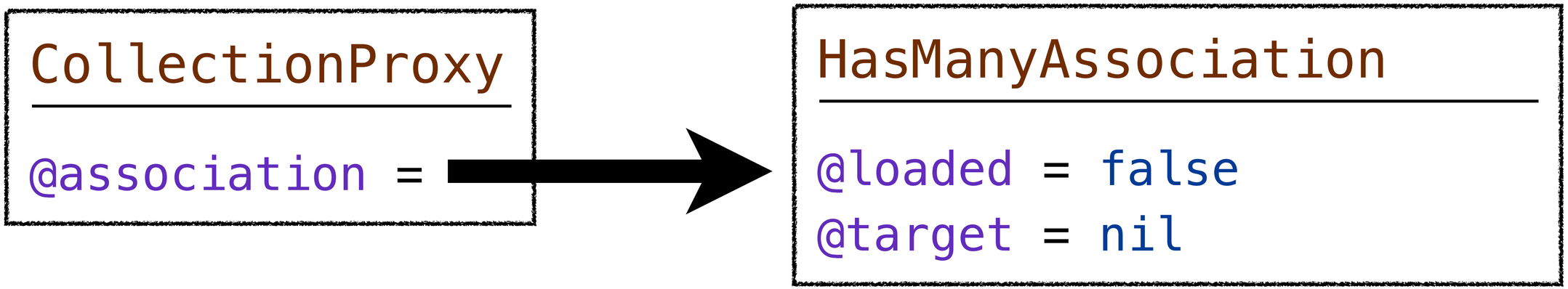
Moved From Reader

Relations



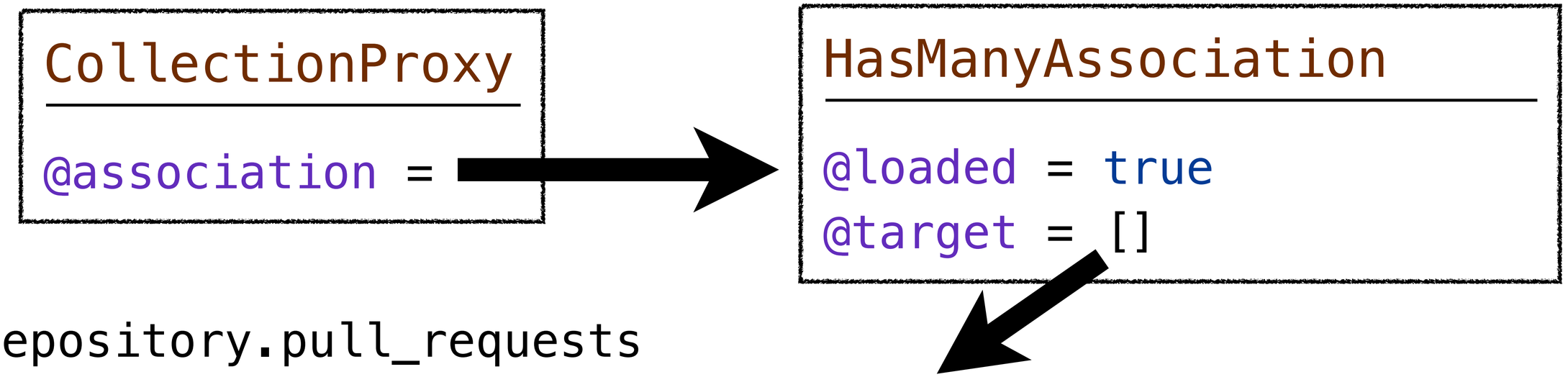

```
repository.pull_requests
```

Relations



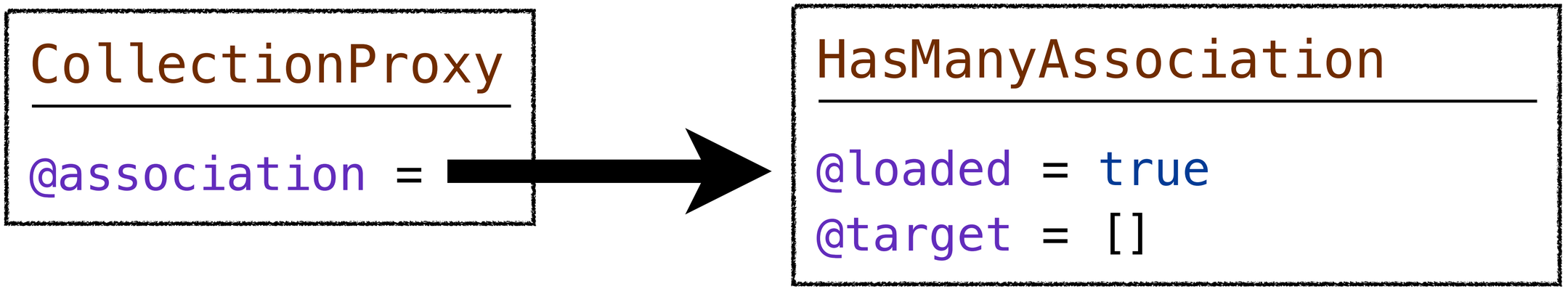

```
repository.pull_requests
repository.pull_requests.to_a
```

Relations



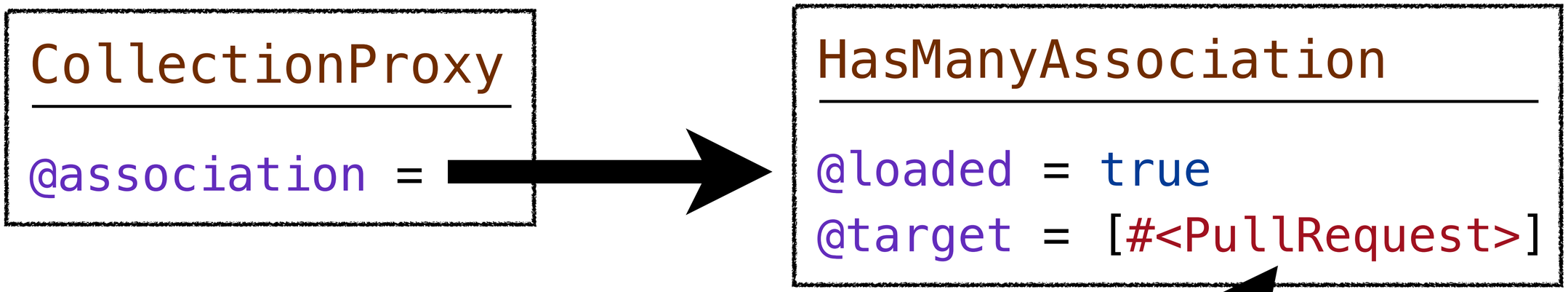

```
repository.pull_requests
repository.pull_requests.to_a => []
```

Relations



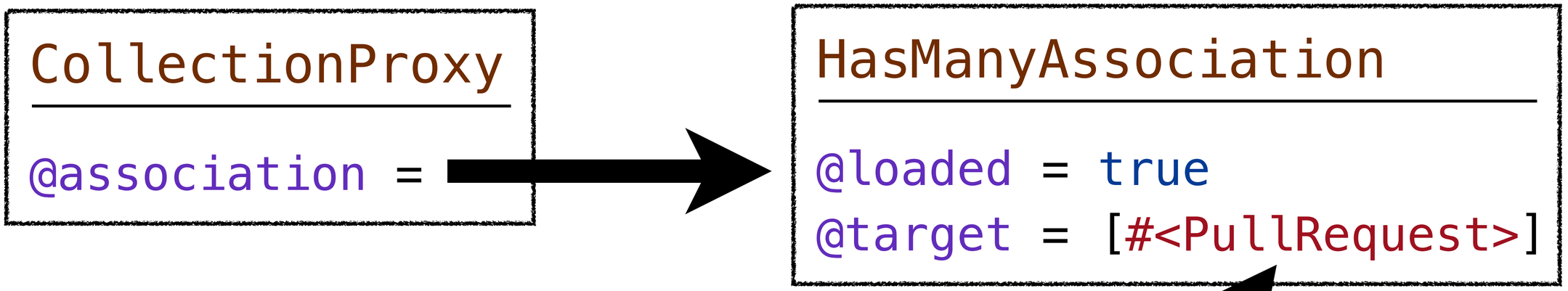

```
repository.pull_requests
repository.pull_requests.to_a => []
repository.pull_requests.create
```

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

```ruby
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
--------------------
@owner      = #<PullRequest>
@loaded     = false
@target     = nil
```

Inverses

```
pull_request.repository
```

```
BelongsToAssociation
--------------------
@owner      = #<PullRequest>
@loaded     = false
@target     = nil
```

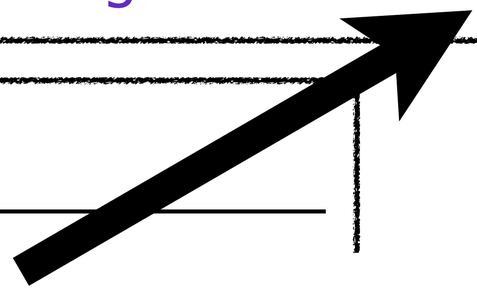
```
HasManyAssociation
------------------
@owner      = #<Repository>
@loaded     = true
@target     = [#<PullRequest>, ...]
```

```
repository.pull_requests
```

Inverses

```
pull_request.repository
```

```
BelongsToAssociation
@owner      = #<PullRequest>
@loaded     = true
@target     = #<Repository>
```



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

```
repository.pull_requests
```

Inverses

```
HasManyAssociation
------------------

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

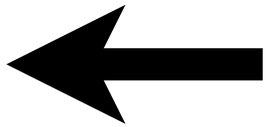
Inverses

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

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

Inverses

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

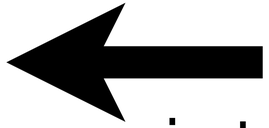


```
#<PullRequest>
```

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

Inverses

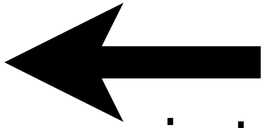
```
@target.each do |record|      #<PullRequest>
  association = record.association(:repository)
```



```
HasManyAssociation
------------------

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

Inverses

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



```
#<PullRequest>
```

```
HasManyAssociation
------------------

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

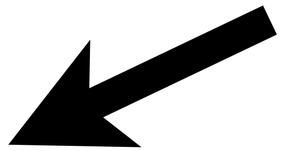
Inverses

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

```
HasManyAssociation
------------------

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

Inverses

Harcoded



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

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
```

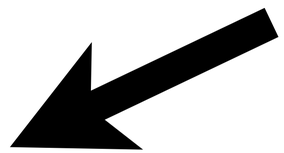
```
#inverse_name => :repository
```

Inverses

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

Inverses

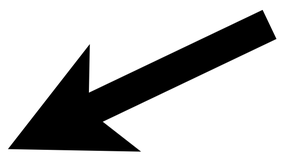
Harcoded



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

Inverses

Generic



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

Further Studies

- Through Associations, Polymorphic Associations, Scoping, Etc.
- github.com/rails/rails
  - activerecord/lib/active_record/associations.rb
  - activerecord/lib/active_record/reflection.rb
  - activerecord/lib/active_record/associations/*
- danieljamescolson.com/blog

Practical Applications

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

RailsConf
Portland

Reflecting on Active Record Associations

Daniel Colson

@composerinteralia

@dodecadaniel