

# Get Help

Sergio Revilla @elreplicante

## http://api.rubyonrails.org/



where(opts = :chain, \*rest)

Returns a new relation, which is the result of filtering the current relation according to the conditions in the argument

where accepts conditions in one of several formats. In the examples below, the resulting SQL is given as an illustra different depending on the database adapter.

#### string

A single string, without additional arguments, is passed to the query constructor as an SQL fragment, and used in t

```
Client.where("orders_count = '2'")
# SELECT * from clients where orders_count = '2';
```

Note that building your own string from user input may expose your application to injection attacks if not done prop recommended to use one of the following methods.

#### array

If an array is passed, then the first element of the array is treated as a template, and the remaining elements are incondition. Active Record takes care of building the query to avoid injection attacks, and will convert from the ruby to needed. Elements are inserted into the string in the order in which they appear.

```
User.where(["name = ? and email = ?", "Joe", "joe@example.com"])
# SELECT * FROM users WHERE name = 'Joe' AND email = 'joe@example.com';
```

Alternatively, you can use named placeholders in the template, and pass a hash as the second element of the arra' replaced with the corresponding values from the hash.

```
User.where(["name = :name and email = :email", { name: "Joe", email: "joe@example.com" }])
# SELECT * FROM users WHERE name = 'Joe' AND email = 'joe@example.com';
```

This can make for more readable code in complex queries.

Lastly, you can use sprintf-style % escapes in the template. This works slightly differently than the previous method that the values in the template are properly quoted. The values are passed to the connector for quoting, but the call enclosed in quotes in the resulting SQL. After quoting, the values are inserted using the same escapes as the Ruby

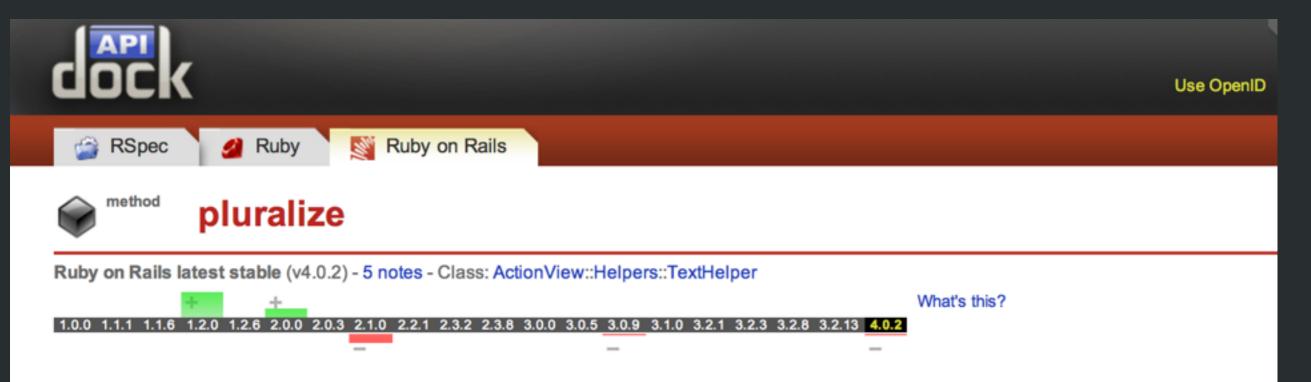
```
User.where(["name = '%s' and email = '%s'", "Joe", "joe@example.com"])
# SELECT * FROM users WHERE name = 'Joe' AND email = 'joe@example.com';
```

If where is called with multiple arguments, these are treated as if they were passed as the elements of a single arra

```
User.where("name = :name and email = :email", { name: "Joe", email: "joe@example.com" })
# SELECT * FROM users WHERE name = 'Joe' AND email = 'joe@example.com';
```



## http://apidock.com/rails



pluralize(count, singular, plural = nil) public

Attempts to pluralize the singular word unless count is 1. If plural is supplied, it will use that when count is > 1, otherwise it will use the Inflector to determine the plural form.

```
pluralize(1, 'person')
# => 1 person

pluralize(2, 'person')
# => 2 people

pluralize(3, 'person', 'users')
# => 3 users

pluralize(0, 'person')
# => 0 people
```

## http://stackoverflow.com/



Questions

Tags

Users

**Badges** 

Unanswered

#### Search

rails

search

#### 226,173 results

relevance

newest

votes

active



#### Q: Understand Rails Authenticity Token

I am running into some issues regarding Authenticity Token in **rails**, as I did many times now. But I really don't want to just solve this problem and go on, I would really like to understand ...

ruby-on-rails

authenticity-token

asked jun 2 '09 by Ricardo Acras

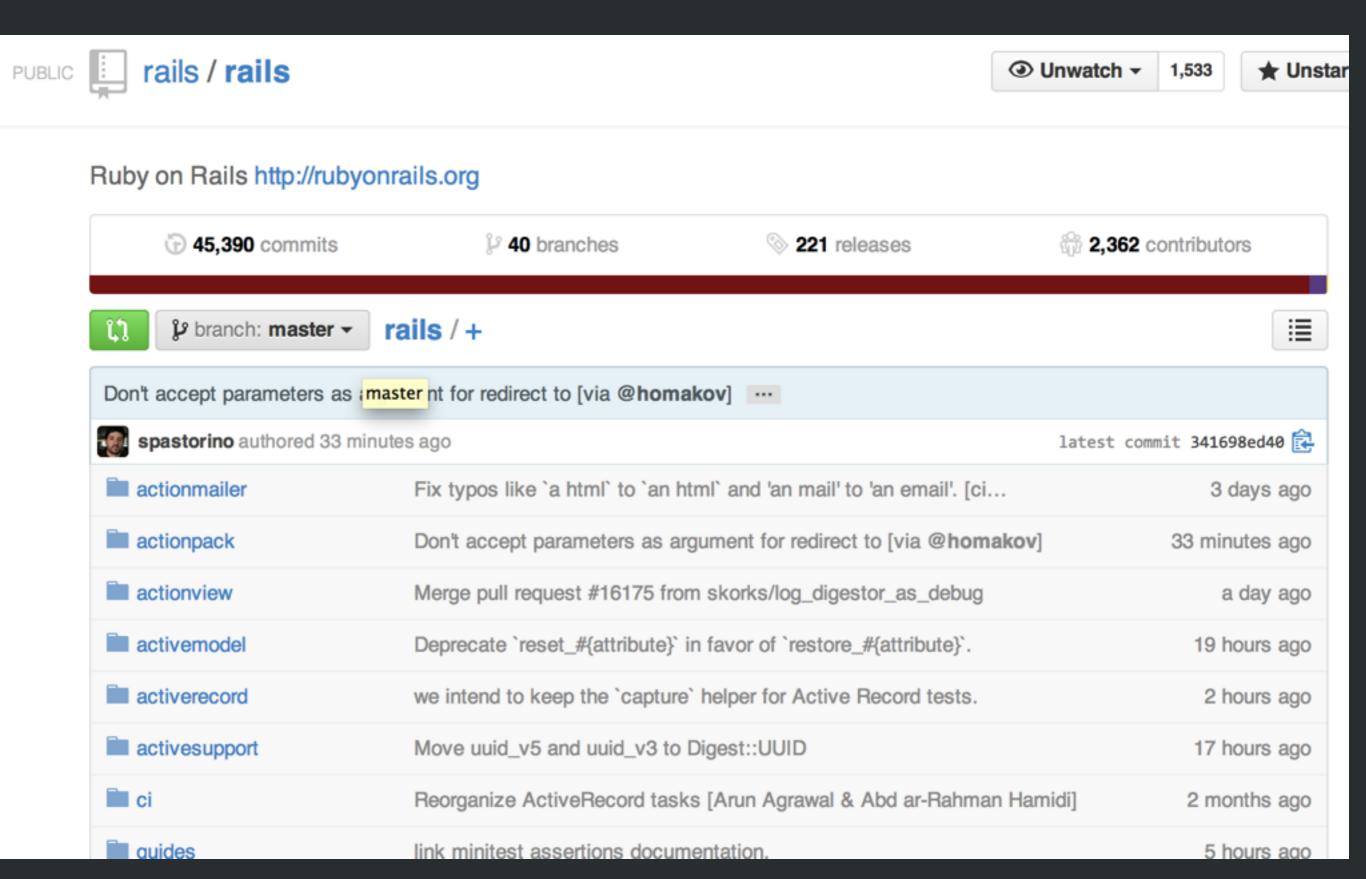


#### A: How do I get the current absolute URL in Ruby on Rails?

\*\*For **Rails** 3.2 or **Rails** 4\*\* You should use request.original\_url to get the current URL. More detail. For **Rails** 3: You want "#{request.protocol}#{request.host\_with\_port}#{request.fullpath}", since ... request.url is now deprecated. For **Rails** 2: You want request.url instead of request.request\_uri. This combines the protocol (usually http://) with the host, and request\_uri to give you the full address. ...

answered jan 29 '10 by Jaime Bellmyer

## http://github.com/rails/rails



# http://ruby-doc.org/

Home Core Std-lib Gems Downloads Search

There was 1 Ruby vulnerability reports in the last 14 days. 1 high. Most recent: CVE-2013-1756. See details.



### Help and documentation for the Ruby programming language.

#### Core API

These are the API documents for the base classes and modules in the current stable release of Ruby 2.1.

- 2.1.2 core Core API docs for Ruby 2.1.2 This is the current official release.
- 2.1.1 core Core API docs for Ruby 2.1.1
- 2.1.0 core Core API docs for Ruby 2.1.0
- more...

#### Standard Library API

These are the API documents for the standard library classes and modules in version 2.1

- The 2.1.2 standard library (this is the current official release)
  - Additional libraries included with the standard Ruby distribution, such as CGI, OpenURI, and REXML
- The 2.1.1 standard library
- more...

#### Getting Started

A collection of resources for those just starting out with

## Docs in Other Languages

French / Français