



developers

- [Hire a developer](#)
- [Apply as a Developer](#)
- [Login](#)
-
- [Top 3%](#)
- [Why](#)
- [Clients](#)
- [Partners](#)
- [Community](#)
- [Blog](#)
- [About Us](#)
- [Hire a developer](#)
- [Apply as a Developer](#)
- [Login](#)
- - Questions?
 - [Contact Us](#)
 - Call us:
 - [+1.888.604.3188](#)
 -
 -
 -
- Questions?
- [Contact Us](#)
- Call us:
- [+1.888.604.3188](#)
-
-
-

[Hire a developer](#)

8 Essential Ruby on Rails Interview Questions*

- 1.1Kshares



-



-



-



-

[Submit an interview question](#)[Submit a question](#)

Looking for experts? Check out Toptal's [Ruby on Rails Developers](#).

What is the difference between Ruby's `Hash` and ActiveSupport's `HashWithIndifferentAccess`?

Hide answer

The `Hash` class in Ruby's core library retrieves values by doing a standard `==` comparison on the keys. This means that a value stored for a `Symbol` key (e.g. `:my_value`) cannot be retrieved using the equivalent `String` (e.g. `'my_value'`). On the other hand, `HashWithIndifferentAccess` treats `Symbol` keys and `String` keys as equivalent so that the following would work:

```
h = HashWithIndifferentAccess.new
h[:my_value] = 'foo'
h['my_value'] #=> will return "foo"
```

[Comment](#)

Fields marked with an asterisk (*) are required

Comment submitted successfully. Thank you.

We are going to review the comment and get back to you as soon as possible.

What's the problem with the following controller code? What would be the consequence of leaving this code in a production app? How would you fix it?

```
class MyController < ApplicationController
  def options
    options = {}
    available_option_keys = [:first_option, :second_option, :third_option]
    all_keys = params.keys.map(&:to_sym)
    set_option_keys = all_keys & available_option_keys
    set_option_keys.each do |key|
      options[key] = params[key]
    end
  end
end
```

```
    options
  end
end
```

Hide answer

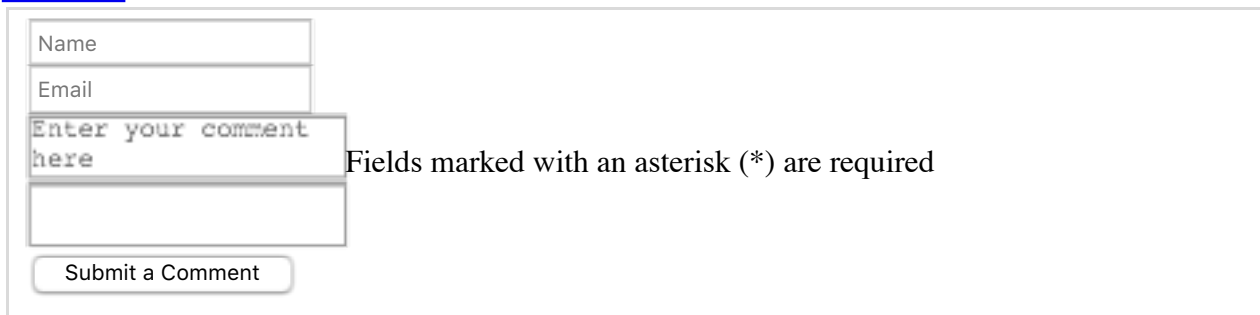
It's dangerous to convert user supplied parameters to symbols, since `symbol` objects in Ruby are not garbage collected. An attacker could send a series of requests with random keys that would be turned into symbols, quickly exhausting your server's available memory and taking down your site.

There are two ways that this could be fixed. The first would be to use `slice` to eliminate values from the `params` hash that are not valid option keys. This would look something like:

```
params.slice(*available_option_keys)
```

An alternative, some would argue better, option would simply be to use `string` keys for your options. Unless you have an extremely large number of possible option keys, you won't actually save that much memory by using `symbol` keys instead.

[Comment](#)



Comment submitted succesfully. Thank you.

We are going to review the comment and get back to you as soon as possible.

What's the issue with the controller code below? How would you fix it?

```
class CommentsController < ApplicationController
  def users_comments
    posts = Post.all
    comments = posts.map(&:comments).flatten
    @user_comments = comments.select do |comment|
      comment.author.username == params[:username]
    end
  end
end
```

Hide answer

This is a classic example of the notorious “n+1” bug. The first line will retrieve all of the `Post` objects from the database, but then the very next line will make an additional request for each `Post` to retrieve the corresponding `Comment` objects. To make matters worse, this code is then making *even more* database requests in order to retrieve the `Author` of each `Comment`.

This can all be avoided by changing the first line in the method to:

```
posts = Post.includes(comments: [:author]).all
```

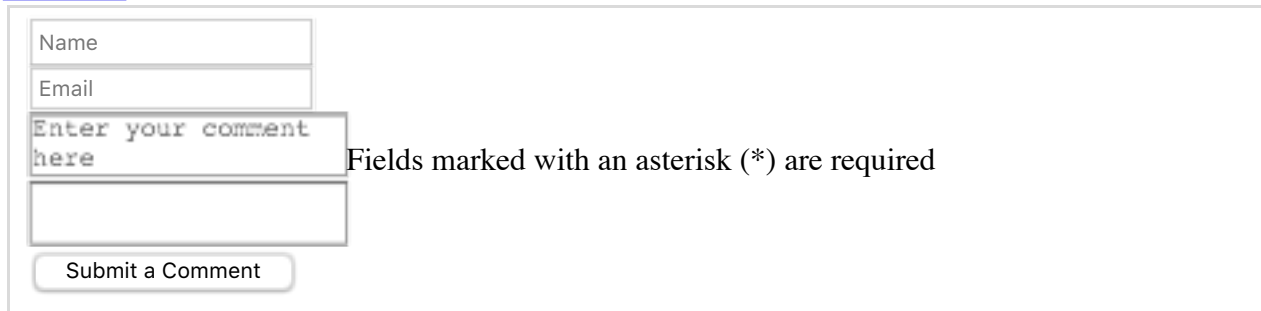
This tells ActiveRecord to retrieve the corresponding `Comment` and `Author` records from the

database immediately after the initial request for all `posts`, thereby reducing the number of database requests to just three.

Please note that the above answer is only one of a few ways that it is possible to avoid incurring an “n+1” penalty, and each alternative will have its own caveats and corner cases. The above answer was selected to be presented here since it requires the smallest change to the existing code and makes no assumptions regarding the reverse association of `Comment` to `Post`.

Incidentally, there’s another issue here (although not what we’re focused on in this question and answer); namely, performing a query in Ruby that could instead be done in the database (and which would very likely be faster there!). A relatively complex query like this can instead be constructed in ActiveRecord pretty easily, thus turning a 3 database query operation (plus some Ruby code executing) into a single database query.

[Comment](#)



Name

Email

Enter your comment here

Submit a Comment

Fields marked with an asterisk (*) are required

Comment submitted successfully. Thank you.

We are going to review the comment and get back to you as soon as possible.

What is CSRF? How does Rails protect against it?

Hide answer

CSRF stands for [Cross-Site Request Forgery](#). This is a form of an attack where the attacker submits a form on your behalf to a different website, potentially causing damage or revealing sensitive information. Since browsers will automatically include cookies for a domain on a request, if you were recently logged in to the target site, the attacker’s request will appear to come from you as a logged-in user (as your session cookie will be sent with the `POST` request).

In order to protect against CSRF attacks, you can add `protect_from_forgery` to your `ApplicationController`. This will then cause Rails to require a CSRF token to be present before accepting any `POST`, `PUT`, or `DELETE` requests. The CSRF token is included as a hidden field in every form created using Rails’ form builders. It is also included as a header in `GET` requests so that other, non-form-based mechanisms for sending a `POST` can use it as well. Attackers are prevented from stealing the CSRF token by browsers’ “same origin” policy.

[Comment](#)

Fields marked with an asterisk (*) are required

Comment submitted succesfully. Thank you.

We are going to review the comment and get back to you as soon as possible.

How would you define a `Person` model so that any `Person` can be assigned as the parent of another `Person` (as demonstrated in the Rails console below)? What columns would you need to define in the migration creating the table for `Person`?

```

irb(main):001:0> john = Person.create(name: "John")
irb(main):002:0> jim = Person.create(name: "Jim", parent: john)
irb(main):003:0> bob = Person.create(name: "Bob", parent: john)
irb(main):004:0> john.children.map(&:name)
=> ["Jim", "Bob"]

```

And for a more advanced challenge: Update the `Person` model so that you can also get a list of all of a person's grandchildren, as illustrated below. Would you need to make any changes to the corresponding table in the database?

```

irb(main):001:0> sally = Person.create(name: "Sally")
irb(main):002:0> sue = Person.create(name: "Sue", parent: sally)
irb(main):003:0> kate = Person.create(name: "Kate", parent: sally)
irb(main):004:0> lisa = Person.create(name: "Lisa", parent: sue)
irb(main):005:0> robin = Person.create(name: "Robin", parent: kate)
irb(main):006:0> donna = Person.create(name: "Donna", parent: kate)
irb(main):007:0> sally.grandchildren.map(&:name)
=> ["Lisa", "Robin", "Donna"]

```

Hide answer

Normally, the target class of an `ActiveRecord` association is inferred from the association's name (a perfect example of "convention over configuration"). It is possible to override this default behavior, though, and specify a different target class. Doing so, it is even possible to have relationships between two objects of the same class.

This is how it is possible to set up a parent-child relationship. The model definition would look like:

```

class Person < ActiveRecord::Base
  belongs_to :parent, class: Person
  has_many :children, class: Person, foreign_key: :parent_id
end

```

It's necessary to specify the `foreign_key` option for the `has_many` relationship because `ActiveRecord` will attempt to use `:person_id` by default. In the migration to create the table for this model, you would need to define, at minimum, a column for the `name` attribute as well as an integer column for `parent_id`.

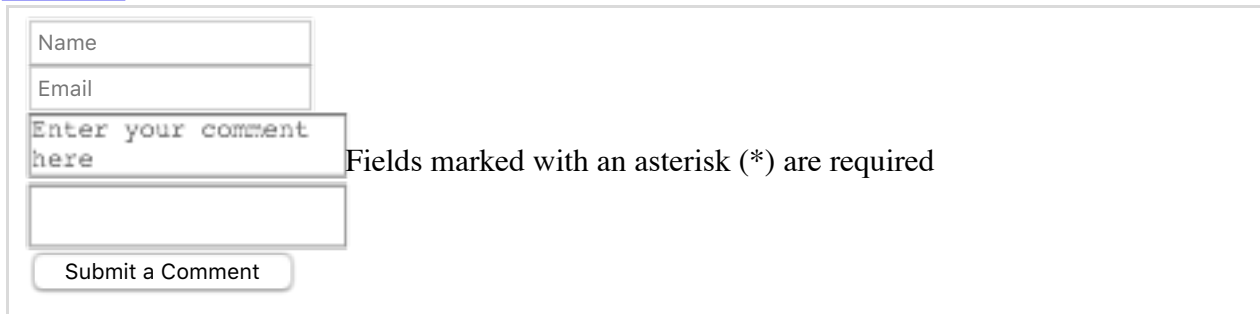
Self-referential relationships can be extended in all the same ways as normal two-model relationships. This even includes `has_many ... :through => ...` style relationships. However,

because we are circumventing Rails' conventions, we will need to specify the source of the `:through` in the case of adding a `grandchild` relationship:

```
class Person < ActiveRecord::Base
  belongs_to :parent, class: Person
  has_many :children, class: Person, foreign_key: :parent_id
  has_many :grandchildren, class: Person, through: :children, source: :children
end
```

Consequently, since this is still just using the `parent_id` defined in the first case, no changes to the table in the database are required.

[Comment](#)



Comment submitted successfully. Thank you.

We are going to review the comment and get back to you as soon as possible.

What paths (HTTP verb and URL) will be defined by the following snippet in `config/routes.rb`?

```
resources :posts do
  member do
    get 'comments'
  end
  collection do
    post 'bulk_upload'
  end
end
```

Hide answer

Using the `resource` method to define routes will automatically generate routes for the standard seven restful actions:

1. GET /posts
2. POST /posts
3. GET /posts/new
4. GET /posts/:id/edit
5. GET /posts/:id
6. PATCH/PUT /posts/:id
7. DELETE /posts/:id

Note that Rails also supports the (relatively) new URL verb `PATCH` for partial updates to records. (In theory, a `PUT` request should only be valid if the entire record is included in the request.)

The extra routes defined inside of the block passed to `resources` will generate one route valid for individual posts (GET /posts/:id/comments) as well as one defined for the top-level resource (POST /posts/bulk_upload).

[Comment](#)

Fields marked with an asterisk (*) are required

Comment submitted succesfully. Thank you.

We are going to review the comment and get back to you as soon as possible.

Create a route to be able to display pages with different information about different types of beer. The route should recognize URL paths like `/beer/<beer_type>` and should use the same controller action for each type of beer with the actually beer type passed into the controller action as a parameter. The valid beer types are:

- IPA
- brown_ale
- pilsner
- lager
- lambic
- hefeweizen

Any other type of beer specified should generate a 404 status code.

Hide answer

One option would be to generate a simple get route that specifies the controller action to call and passes the kind of beer as a parameter:

```
get 'beers/:kind' => 'beers#kind'
```

Then, within the context of the controller action, if the `kind` parameter is not included in the list of valid kinds, the action can raise a `ActionController::RoutingError`, which will redirect to 404 in production.

Alternatively, *a simpler solution* is to check against the list of valid kinds in the definition of the route. This can be accomplished using the `constraints` option as follows:

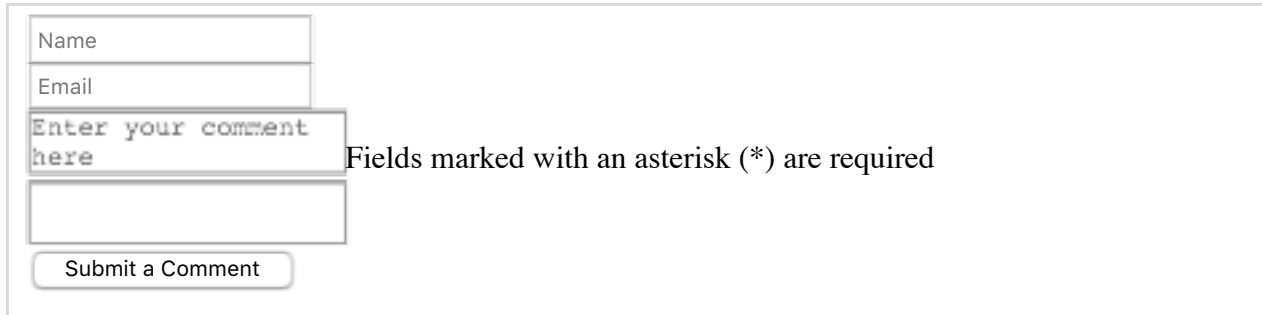
```
kinds = %w|IPA brown_ale pilsner lager lambic hefeweizen|
get 'beers/:kind' => 'beers#kind', constraints: {kind: Regexp.new(kinds.join('|'))}
```

This code calls the `BeersController#kind` action method with `params['kind']` set to a string representing the beer type given in the URL path. The key is using the `constraints` option for the route to specify a regular expression to use to verify the route is correct. In this case, the lambda checks to see that the `kind` parameter is included in the list of valid beer types.

Or perhaps *an even better solution* would be to use resource routing. This has the added benefit of providing URL generation helpers, but at the cost of requiring that the parameter name for the beer be passed as `:id`. This would look something like:

```
kinds = %w|IPA brown_ale pilsner lager lambic hefweizen|
resource :beer, only: [:show], constraints: {id: Regexp.new(kinds.join('|'))}
```

[Comment](#)



Comment submitted succesfully. Thank you.

We are going to review the comment and get back to you as soon as possible.

Suppose we have a Student with id="4". If we delete the Student with id="4", what will be the result of each of the following queries:

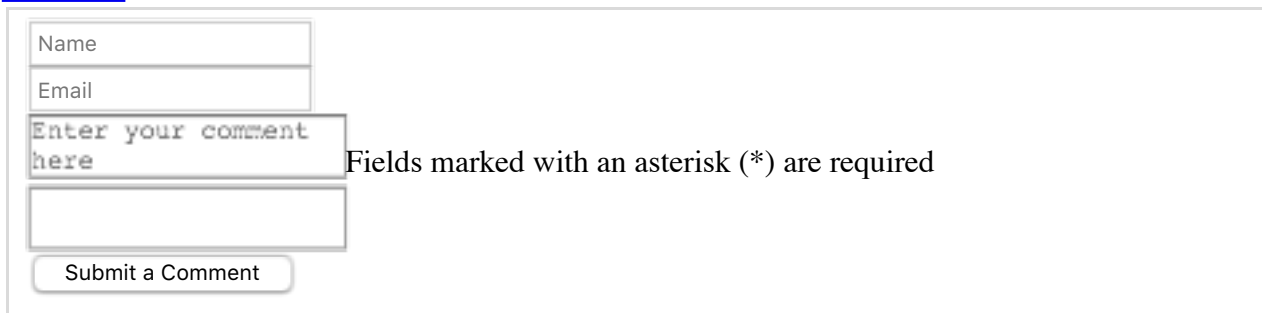
```
Student.find(4)
```

```
Student.find_by_id(4)
```

Hide answer

`Student.find(4)` will raise an error `ActiveRecord::RecordNotFound: Couldn't find User with id=4` whereas `Student.find_by_id(4)` will not raise an error and will return `nil`.

[Comment](#)



Comment submitted succesfully. Thank you.

We are going to review the comment and get back to you as soon as possible.

* There is more to interviewing than tricky technical questions, so these are intended merely as a guide. Not every "A" candidate worth hiring will be able to answer them all, nor does answering them all guarantee an "A" candidate. At the end of the day, [hiring remains an art, a science — and a lot of work](#).

Submit an interview question

Submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Name

Email

Enter your question here

Enter your answer here

All fields are required

☐ I agree with the Terms and Conditions of Toptal, LLC's [Privacy Policy](#)

Submit a Question

Thanks for submitting your question.

Our editorial staff will review it shortly. Please note that submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Looking for Ruby on Rails experts? Check out Toptal's [Ruby on Rails developers](#).



[View full profile »](#)

[Graham Powrie](#)

United States

Graham is a designer and developer focused on creating custom web applications that automate and improve business processes, while remaining flexible for iteration. He's a full-stack developer who can deliver a complet...

[Ruby on RailsCoffeeScriptJavaScriptRubyEmber.js](#)

[Hire Graham](#)



[View full profile »](#)

[Eqbal Quran](#)

Jordan

Eqbal is a senior full-stack developer with more than a decade of experience working in web and mobile development. He is a masterful problem solver, and boasts an extensive portfolio of finished professional products.

[Ruby on Rails](#)[Ruby](#)[GitHub](#)+2 more

[Hire Eqbal](#)



[View full profile »](#)

[Slobodan Kovacevic](#)

Serbia

Slobodan is a senior web developer and programmer with over a decade of experience working on the web. He is passionate about Ruby and Ruby on Rails. He can work in different roles, ranging from an individual developer to technic...

[Ruby on Rails](#)[SQL](#)[Ruby](#)[SQLite](#)[GitHub](#)

[Hire Slobodan](#)



Are you hiring [Ruby on Rails developers](#)? We'll find you a great freelance Ruby on Rails developer to join your team.

Alvaro Oliveira

VP of Talent Operations

[Hire a top Ruby on Rails developer now](#)

Toptal connects the [top 3%](#) of freelance designers and developers all over the world.

Join the Toptal community.

[Hire a developer](#)

or

[Apply as a Developer](#)

Highest In-Demand Talent

- [iOS Developer](#)
- [Java Developer](#)
- [.NET Developer](#)
- [Front-End Developer](#)
- [UX Designer](#)
- [UI Designer](#)

About

- [Top 3%](#)
- [Clients](#)
- [Freelance developers](#)
- [Freelance designers](#)
- [About Us](#)

Contact

- [Contact Us](#)
- [Press Center](#)
- [Careers](#)

- [FAQ](#)

Social

- [Facebook](#)
- [Twitter](#)
- [Google+](#)
- [LinkedIn](#)

[Toptal](#)

Hire the top 3% of freelance talent

- © Copyright 2010 - 2016 Toptal, LLC
- [Privacy Policy](#)
- [Website Terms](#)

[Home](#) › [Ruby on Rails](#) › [Interview questions](#)

-
- [Top 3%](#)
- [Why](#)
- [Clients](#)
- [Partners](#)
- [Community](#)
- [Blog](#)
- [About Us](#)
- [Hire a developer](#)
- [Apply as a Developer](#)
- [Login](#)
- - Questions?
 - [Contact Us](#)
 - Call us:
 - [+1.888.604.3188](#)
 -
 -
 -
- Questions?
- [Contact Us](#)
- Call us:
- [+1.888.604.3188](#)
-
-
-

[Hire a developer](#)