Nsikan Sylvester FOLLOW
Full Stack Engr. JavaScript, Ruby, C++, and Rust

# Guide on Acing Ruby on Rails Technical Interviews

Published Nov 06, 2022 Last updated Dec 25, 2023


## INTRODUCTION

If you have ever stepped forward to apply for positions higher than your current level or just want to change your current job, there are chances that you will get frustrated failing technical interviews. If you have the required expertise and experience but still find yourself not making it through, you do not have to feel bad. Although the interviewer judged you based on the outcome of your interview. You are not alone in this. So many developers faced this problem, not because they are not fit for the job, but just could not make it through at that moment

So many posts give pasts interview questions, you must have read a lot of them but were not asked any of those in your interview. However, you could have aced it if you knew the topics that would be asked.

If you ever felt like giving up, please do not do it. "DO NOT GIVE UP", brace up, you do make a difference. You could be just a piece of information away from acing your next Ruby on rails interview.
This post aims to enumerate the areas that are often asked in a Ruby on rails interview including advanced concepts on Ruby/Ruby on Rails, which would get you grounded for your technical interview.
Go over this post before you attempt any Ruby on Rails Technical Interview.
And feel free to state any additional heading that could help a Ruby on rails Engineer in a technical interview.

## 1. The Ruby Truthy and Equality Operators

In ruby false and nil evaluates to false every other thing is true.
Read more here: truthiness-in-ruby

In ruby equality, several operators work differently. Have at your fingertips the difference between ==, eql?, equal? and === Operators.
== test the difference between two objects
eql is used by hash to test for the same hash key.
Equal? Checks if two operands refer to the same object.
=== The case equality operator checks if the object on the right-hand side is a member of the object on the left-hand side.

Read more here: difference-between-eql-equal-in-ruby

## 2.Object Oriented Programming.

You have heard that in ruby everything is an object. This is the Object paradigm. which means that in the Ruby language, everything is an object. These objects, regardless of whether they are strings, numbers, classes, modules, etc, operate in a system called The Object Model. Read more here...

1. why object oriented programming
2. ruby-object-model

## 3.Code Quality

Be passionate about writing Clean Ruby Code.
Get acquainted with ruby/rails code smell detection tools.
Examples are: reek, rubocop, rails best practices, brakeman etc.
Read more in the following links:

1. ruby-on-rails-code-optimization-and-cleanup
2. check-my-code-tips-to-keep-ruby-codes-smell-free
3. Reek: Code Smell Detector for Ruby

## 4. Know and apply appropriate Design Patterns

SOLID design is a good place to start. Also, Creational, Behavioural, and Structural design patterns are worth reviewing.
This not only helps understand code written by other engineers, it gives you a sense of an understanding of suitable approaches to solving a problem.

1. ruby-solid-design-principles
2. design-patterns
   View the ruby code example on each pattern as follows

## 5. Mocking: Doubles, Stubs, mock, and Spies

Practicing TDD/BDD by just writing simple test cases is not enough, Candidates are required to possess the necessary skills for mocking. (test doubles, stubs, mock, and spies).

Below are definitions of these concepts:

“A test double is a simplified version of an object that allows us to define “fake” methods and their return values.”

“A method stub is an implementation that returns a pre-determined value. ”

“A mock is a stub with a built-in expectation to be satisfied during the test. ”

“Spies are objects that by default can accept all methods without throwing any exception and keep history of the methods called on them. ”

To see them in action read the following posts.

mocking-in-ruby-with-minitest
the-difference-between-mocks-stubs-and-spies

## 6. Modules and Mixins

Ruby supports single inheritance, meaning that a class can only inherit from a single class. Modules and mixins play an important role in overcoming this restriction in favor of composition over inheritance. It enables the application to share code in other places.

Take note of 3 ways to import a module to a class.

**1. via include keyword**

When importing a module with an include statement, the module is inserted into the ancestor chain of the class importing the module. The module methods can be accessed via the class's instance.

**2. via extend keyword**

When importing a module with an extend statement, the module methods are imported as class methods.

**3. via prepend keyword**

It works similar to include, the module gets inserted at the bottom of the chain even before the class. Method calls on class instance are first looked up in the module methods before the class methods.
Read more here:

1. ruby-modules-include-vs-prepend-vs-extend
2. using-mixins-and-modules-in-your-ruby-on-rails-application

## 7. Rails Security: CSRF Token

CSRF token is what rails use to prevent Cross-site request forgery attacks. Rails compare the token from the page with the token from the session cookie to ensure they match.

As a Rails developer, you basically get CSRF protection for free. It starts with this single line in **application_controller.rb**, which enables CSRF protection:

```
protect_from_forgery with: :exception
```

```
<%= csrf_meta_tags %>
```

Read more here a-deep-dive-into-csrf-protection-in-rails

## 8. Rails API only application and Use of Third Party API

Using rails to build a back-end that is shared between web and native applications.
Read the following links:

1. beginners-guide-to-building-a-rails-api
2. binar-academy/simple-rails-api-server-using-simple-tdd

## 9. Lazy Loading vs Eager Loading in Rails when to use

1. what-is-lazy-loading-in-rails
2. improving-the-performance-of-your-rails-app-with-eager-loading
3. better-performance-for-rails-app-with-joins-and-eager-loading
4. joins-vs-preload-vs-includes-vs-eager-load-in-rails

## 10 Rails Code Quality Tools

top-8-tools-for-ruby-on-rails-code-optimization-and-cleanup

## 11. Latest features in Rails

## 12. Activerecord Association and Self join

ruby-on-rails-active-record-associations

## 13. Service Oriented Architecture

1. service-oriented-architecture-rails
2. building-service-oriented-architecture-using-rails-and-kafka

## 14. Implementing Websockets in Ruby on Rails (Action Cable)

ruby-on-rails-websockets-and-actioncable

## 15. Classes and Service Objects

A service object in Rails is a plain old Ruby object created for a specific business action. This is code written outside of the model, controller, or view layer. It could be making a POST request to post a transaction to a third-party API.

1. using-service-objects-in-ruby-on-rails
2. refactor-ruby-rails-service-object

## 16. Actionmailer and Emailing in rails

sending-emails-in-rails-with-action-mailer-and-gmail

## 17. CI/CD and Cloud Deployment

Continuous Integration (CI) pipeline is the norm now for modern applications. Many CI/CD pipeline technologies exist, and a developer is expected to be

conversant with a few of them.
Below are tutorials on CI/CD pipelines.

1. rails-continuous-integration
2. set-up-ci-cd-for-rails-app-using-github-actions-aws-beanstalk
3. setting-up-a-rails-app-for-codebuild-codedeploy-and-codepipeline-on-aws

## 18. Rack Application

Rack is the underlying technology behind nearly all of the web frameworks in the Ruby world.
The rack provides a minimal, modular, and adaptable interface for developing web applications in Ruby.
Rack is used by Rails and Sinatra. You can learn more about it here:

1. rack technology
2. definitive-guide-to-rack

## 19. Problem Solving - Algorithm and Data Structure

This is like a constant after a chat every interviewer could give you a problem to solve. No matter how little it may be. I recommend practicing with: leetcode Questions
and Hacker rank Hacker rank interview-preparation-kit

## 20. Advanced ActiveRecord Querying and Caching.

1. advanced-activerecord-querying
2. executing-an-sql-query-directly-from-activerecord

Caching:

1. rails-low-level-caching
2. ruby-on-rails-cachings

## Summary

Preparation is paramount to ace any technical interview. This post presents critical areas that have been carefully selected to have at your fingertips before approaching a Ruby Technical interview. It is just a guide to help to refresh your memory in areas that may easily skip your mind.

Ruby on Rails | Ruby | Interview preparation | Interview Skills | Rails api