Ruby and Ruby on Rails have recently emerged as some of the most sought after skills in  web development. Even if you’re a general or junior-developer, you’ll most likely be expected to have some knowledge and experience with the ruby language and rails framework. The average Ruby developer salary is currently around $80k/year, while particularly skilled and senior developers can fetch almost twice that. Furthermore, many of the web’s most popular sites, including Hulu, Github, Twitter, Groupon, Yellow Pages, are built on the rails framework. Set against this backdrop, it’s no wonder there’s been an explosion of interest in learning ruby.

Keeping that in mind, below is a list of ruby interview questions from around the web that you should run through before your upcoming developer interview to get your mind running. This list is by no means exhaustive, and prospective employees should be prepared to handle much more complex questions and exercises than these.

Remember that the best interview questions are open ended and have no definitive answers –  a  good interviewer knows this and will be looking for your thought process and problem solving skills, rather than the ‘correct’ answer. That being said, there are still some straight forward questions that any Rails programmer should be able to answer and discuss.  Yes, many of these questions are relatively simple but being able to explain them in plain english may be more difficult than you think. To quote Albert Einstein, ‘If you can’t explain it simply, you don’t understand it well enough’

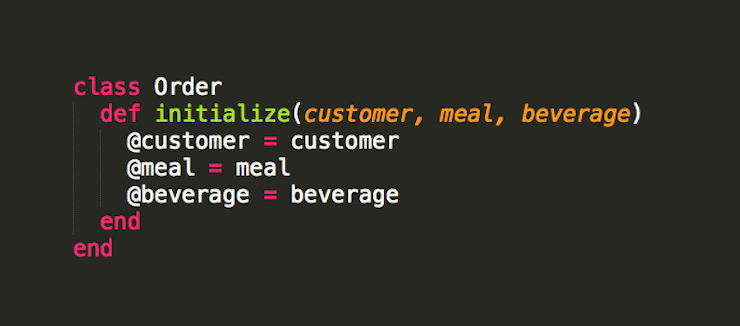
**Open Ended Questions**

1. Explain how (almost) everything is an object in Ruby:
   * This is a simple question based on complex concept. Here’s your chance to show off your theoretical knowledge and demonstrate that you can have an in depth conversation on class hierarchies, inheritance, methods, encapsulation, polymorphism, and more.
   * Explaining this could take an hour or a few minutes – there’s no single correct answer
   * here, save from being able to demonstrate your familiarity with OOP concepts.
2. What’s your favorite testing tool?
   * The specific answer here is probably not important in and of itself – What’s important is that you can demonstrate familiarity with at least several testing tools, and be able to discuss their individual advantages and weaknesses. Never ventured outside of Rails default testing tools? Take some time to familiarize yourself with tools such as Rspec, FactoryGirl, Capybara, and Cucumber.
3. What are Gems and which are some of your favorites?
   * Short answer: Gems are packaged bits of Ruby code that you can install to extend or add functionality to your app.
   * Be sure to be able to discuss a list of your favorite gems, why you like them, and any customizations you like to add. This is also a good opportunity to highlight any gems you may have published.

Shift into overdrive with these top recommended courses.

**General Knowledge**

1. What is a class?
   * You should easily be able to explain not only what a class is, but how and when you would create a new one as well as what functionality it would provide in the larger context of your program.
2. What is the difference between a class and a module?
   * The straightforward answer: A module cannot be subclassed or instantiated, and modules can implement mixins.
   * Be prepared to discuss what this actually means in real life, and when you would use a module vs. a class and why.
3. What is an object?
   * Textbook answer here is that an object is an instance of a class and has state, behavior, and identity. In a plain text example, you can say that a truck and a car are both objects of the class Vehicle, or that apple and pear are both objects of the class Fruit.
   * You should be able to explain in detail how object structure and behavior relate to their common class, and why this is so important in Ruby (see question 1).
4. How would you declare and use a constructor in Ruby?
   * Constructors are declared via the initialize method and get called when you call on a new object to be created.
   * Using the code snippet below, calling Order.new acts as a constructor for an object of the class Order.

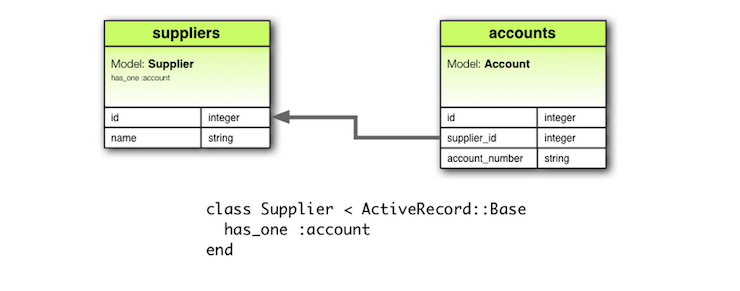


Initializing instance variables of the class Order.

1. How does a symbol differ from a string?
   * Short answer: symbols are immutable and reusable, retaining the same object\_id.
   * Be prepared to discuss the benefits of using symbols vs. strings, the effect on memory usage, and in which situations you would use one over the other.
2. How and when would you declare a Global Variable?
   * Global variables are declared with the ‘$’ symbol and can be declared and used anywhere within your program. You should use them sparingly to never.
3. How would you create getter and setter methods in Ruby?
   * Setter and getter methods in Ruby are generated with the attr\_accessor method. attr\_accessor is used to generate instance variables for data that’s not stored in your database column.
   * You can also take the long route and create them manually.
4. Describe the difference between class and instance variables?
   * Class variables are created with the prefix ‘@@’ and are shared by all objects in a class.
   * Instance variables are created with the prefix ‘@’ and belong to a single object within a class.
   * Beyond the simple textbook definition, be able to describe an example of a class and how you would use class and instance variables within it, and how they relate to issues of class inheritance.
5. Explain some of the looping structures available in Ruby?
   * For loop, While loop, Until Loop.
   * Be able to explain situations in which you would use one over another.

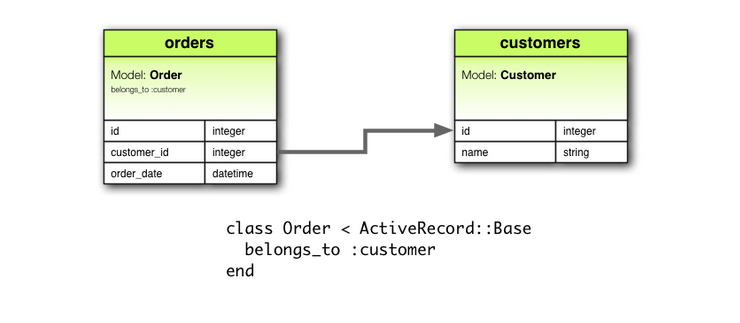
**Model Associations**

1. Explain the difference between a has\_one and belongs\_to association:
   * **has\_one:** Indicates a direct 1:1 relationship between objects where each instance of a model contains one instance of another model.
     + - A product **has\_one** provider, a customer **has\_one** order.



A has\_one association schematic, from Rubyonrails.org

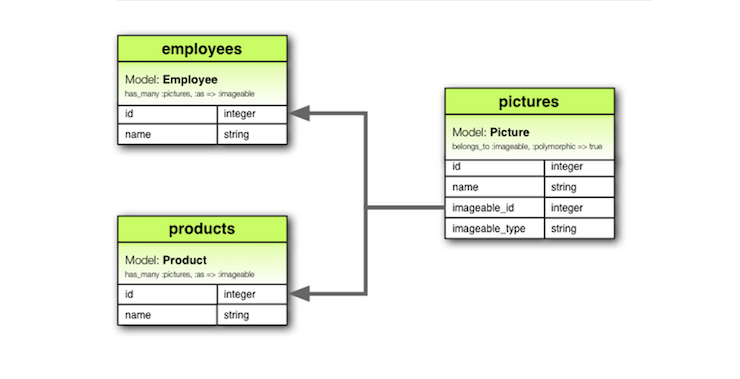
* **belongs\_to:** Represents the inverse of a has\_one (or has\_many) association.
  + ex) An order **belongs\_to**  a customer.



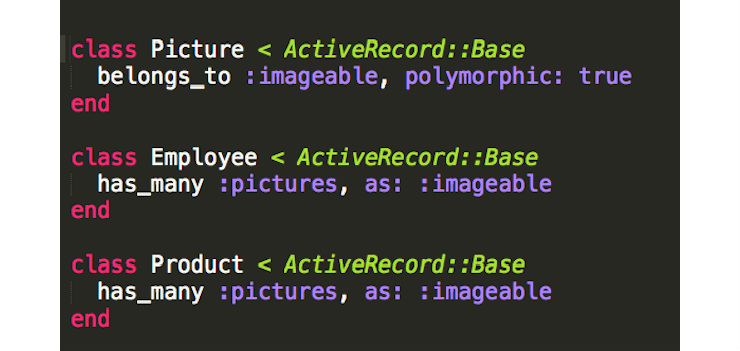
A belongs\_to association schematic, from Rubyonrails.org

* + A good way to remember this is that if a table has foreign keys, its model should have a belongs\_to association.
  + Be able to explain the difference and describe an example of how you would assign these associations to two related models.

1. Explain a polymorphic association:
   * Polymorphic associations allow a model to belong to more than one other model through a single association.



A polymorphic association schematic, from Rubyonrails.org

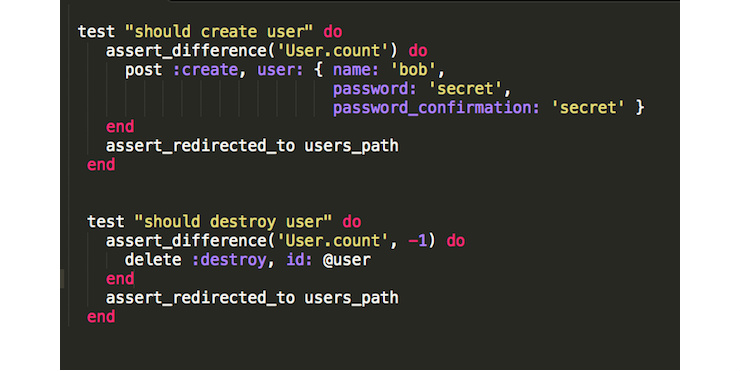


Model structures of a polymorphic association, in Ruby code, from Ruybonrails.org

* + Here, the class Picture **belongs\_to** both Employee and Product, but does so through a single association rather than through multiple.
  + Be sure to know an appropriate situation to create a polymorphic association, such as creating a comment model associated with multiple other models (articles, photos, etc.). The advantage of using polymorphism here is that it allows you to create a single comment model, rather than separate models for each one (PhotoComment model, ArticleComment model, etc.)

**Procs, Lambdas, and Methods**

1. What is a Proc?
   * Procs, short for procedures, act similar to blocks, but can be saved as variables and reused. Think of them as blocks you can call over and over again on multiple arrays.
   * Explain an instance when you would use a proc over a block. [Read more here](http://www.reactive.io/tips/2008/12/21/understanding-ruby-blocks-procs-and-lambdas/)
2. What is a lambda?
   * Lambdas are very similar to procs in terms of functionality. However, they have a few key differences. Lambdas check the number of arguments passed and will return an error if you try to pass the wrong number (while procs set extra variables to nil). The other difference is that lambdas can handle a return function, whereas procs will return an error. [Read more here](http://www.reactive.io/tips/2008/12/21/understanding-ruby-blocks-procs-and-lambdas/)
3. What are the three levels of method access control for classes and what do they signify? What do they imply about the method?
   * Public, protected, and private.
   * Public methods can be called by all objects and subclasses of the class in which they are defined in.
   * Protected methods are only accessible to objects within the same class.
   * Private methods are only accessible within the same instance.
   * Be able to explain why this does (or doesn’t matter), and when you would want to set a method as private.
4. Explain what functional testing is:
   * Functional testing in Rails allows you to test the response of  various actions contained in a controller. Using the Rails default test library, mini test, functional tests use a collection of assert statements that will tell your testing library to expect a certain response based on a control request passed in (either a get, post, patch, put, head, delete request).
   * The two example tests below show functional tests for making sure the post and delete requests in our UsersController properly create and destroy users. The functional tests do this by making sure the requests result in a change in the User.count and that they then redirect to the desired pages.



Functional tests for the create and destroy methods in a User model.

1. Other helpful tips:
   * Aside from being able to demonstrate proficient knowledge, you should also be able to demonstrate your enthusiasm for RoR by highlighting your individual contributions to the community. Be sure to be able to share and discuss your favorite Ruby or Rails blogs, your positions on contested topics within the Rails community, or any open source contributions you’ve made. This is also your opportunity to show off any independent projects you’ve been working on or any Gems you’ve published.

**Ace Your Ruby Interview**

While it’s great to be able to speak to all of these concepts and show that you can carry an in-depth technical conversation, there’s probably nothing more valuable than having a solid portfolio of apps and a demonstrable level of skill.  Being able to speak to these issues is important, and verbal communication skills are highly indicative of future job performance. However, showing that you can apply and manipulate these concepts is much more difficult than speaking to them in generalities. If you’re already a great RoR developer, most of these topics will be like second nature to you. If you’re still a junior-level developer, spend some time not only reviewing and understanding these concepts, but also making sure you can properly utilize them in a real world setting.

# [Top 34 Ruby on Rails Interview Questions](http://career.guru99.com/top-34-ruby-on-rail-interview-questions/)

**1) Explain what is Ruby on Rails?**

* **Ruby:** It is an object oriented [programming](http://career.guru99.com/category/programming-2/) language inspired by PERL and PYTHON.
* **Rails:** It is a framework used for building web application

**2) Explain what is class libraries in Ruby?**

Class libraries in Ruby consist of a variety of domains, such as data types, thread programming, various domains, etc.

**3) Mention what is the naming convention in Rails?**

* **Variables:** For declaring Variables, all letters are lowercase, and words are separated by underscores
* **Class and Module:** Modules and Classes uses MixedCase and have no underscore; each word starts with a uppercase letter
* [**Database**](http://career.guru99.com/category/database/) **Table:** The database table name should have lowercase letters and underscore between words, and all table names should be in the plural form for example invoice\_items
* **Model:** It is represented by unbroken MixedCase and always have singular with the table name
* **Controller:** Controller class names are represented in plural form, such that OrdersController would be the controller for the order table.

**4) Explain what is “Yield” in Ruby on Rails?**

A Ruby method that receives a code block invokes it by calling it with the “Yield”.

**5) Explain what is ORM (Object-Relationship-Model) in Rails?**

ORM or Object Relationship Model in Rails indicate that your classes are mapped to the table in the database, and objects are directly mapped to the rows in the table.

**6) Mention what the difference is between false and nil in Ruby?**

In Ruby False indicates a Boolean datatype, while Nil is not a data type, it have an object\_id 4.

**7) Mention what are the positive aspects of Rails?**

Rails provides many features like

* **Meta-programming:** Rails uses code generation but for heavy lifting it relies on meta-programming. Ruby is considered as one of the best language for Meta-programming.
* **Active Record:** It saves object to the database through Active Record Framework. The Rails version of Active Record identifies the column in a schema and automatically binds them to your domain objects using metaprogramming
* **Scaffolding:** Rails have an ability to create scaffolding or temporary code automatically
* **Convention over configuration:** Unlike other development framework, Rails does not require much configuration, if you follow the naming convention carefully
* **Three environments:** Rails comes with three default environment [testing](http://career.guru99.com/category/testing/), development, and production.
* **Built-in-testing:** It supports code called harness and fixtures that make test cases to write and execute.

**8) Explain what is the role of sub-directory app/controllers and app/helpers?**

* App/controllers: A web request from the user is handled by the Controller. The controller sub-directory is where Rails looks to find controller classes
* App/helpers: The helper’s sub-directory holds any helper classes used to assist the view, model and controller classes.

**9) Mention what is the difference between String and Symbol?**

They both act in the same way only they differ in their behaviors which are opposite to each other. The difference lies in the object\_id, memory and process tune when they are used together. Symbol belongs to the category of immutable objects whereas Strings are considered as mutable objects.

**10) Explain how Symbol is different from variables?**

Symbol is different from variables in following aspects

* It is more like a string than variable
* In Ruby string is mutable but a Symbol is immutable
* Only one copy of the symbol requires to be created
* Symbols are often used as the corresponding to enums in Ruby

**11) Explain what is Rails Active Record in Ruby on Rails?**

Rails active record is the Object/Relational Mapping (ORM) layer supplied with Rails. It follows the standard ORM model as

* Table map to classes
* Rows map to objects
* Columns map to object attributes

**12) Explain how Rails implements Ajax?**

Ajax powered web page retrieves the web page from the server which is new or changed unlike other web-page where you have to refresh the page to get the latest information.

Rails triggers an Ajax Operation in following ways

* **Some trigger fires:** The trigger could be a user clicking on a link or button, the users inducing changes to the data in the field or on a form
* **Web client calls the server:** A Java-script method, XMLHttpRequest, sends data linked with the trigger to an action handler on the server. The data might be the ID of a checkbox, the whole form or the text in the entry field
* **Server does process:** The server side action handler does something with the data and retrieves an HTML fragment to the web client
* **Client receives the response:** The client side JavaScript, which Rails generates automatically, receives the HTML fragment and uses it to update a particular part of the current

**13) Mention how you can create a controller for subject?**

To create a controller for subject you can use the following command

C:rubylibrary> ruby script/generate controller subject

**14) Mention what is Rails Migration?**

Rails Migration enables Ruby to make changes to the database schema, making it possible to use a version control system to leave things synchronized with the actual code.

**15) List out what can Rails Migration do?**

Rails Migration can do following things

* Create table
* Drop table
* Rename table
* Add column
* Rename column
* Change column
* Remove column and so on

**16) Mention what is the command to create a migration?**

To create migration command includes

C:rubyapplication>ruby script/generate migration table\_name

**17) Explain when self.up and self.down method is used?**

When migrating to a new version, **self.up** method is used while **self.down** method is used to roll back my changes if needed.

**18) Mention what is the role of Rails Controller?**

The Rails controller is the logical center of the application. It faciliates the interaction between the users, views, and the model. It also performs other activities like

* It is capable of routing external requests to internal actions. It handles URL extremely well
* It regulates helper modules, which extend the capabilities of the view templates without bulking of their code
* It regulates sessions; that gives users the impression of an ongoing interaction with our applications

**19) Mention what is the difference between Active support’s “HashWithIndifferent” and Ruby’s “Hash” ?**

The **Hash** class in Ruby’s core library returns value by using a standard **“= =”** comparison on the keys. It means that the value stored for a **symbol** key cannot be retrieved using the equivalent string. While the **HashWithIndifferentAccess** treats Symbol keys and String keys as equivalent.

**20) Explain what is Cross-Site Request Forgery (CSRF) and how Rails is protected against it?**

CSRF is a form of attack where hacker submits a page request on your behalf to a different website, causing damage or revealing your sensitive data. To protect from CSRF attacks, you have to add **“protect\_from\_forgery”** to your **ApplicationController**. This will cause Rails to require a CSRF token to process the request. CSRF token is given as a hidden field in every form created using Rails form builders.

**21) Explain what is Mixin in Rails?**

Mixin in Ruby offers an alternative to multiple inheritances, using mixin modules can be imported inside other class.

**22) Explain how you define Instance Variable, Global Variable and Class Variable in Ruby?**

* Ruby Instance variable begins with — **@**
* Ruby Class variables begin with — **@@**
* Ruby Global variables begin with — **$**

**23) Explain how you can run Rails application without creating databases?**

You can execute your application by uncommenting the line in environment.rb

path=> rootpath conf/environment.rb

config.frameworks = [ action\_web\_service, :action\_mailer, :active\_record]

**24) Mention what is the difference between the Observers and Callbacks in Ruby on Rails?**

* **Rails Observers:** Observers is same as Callback, but it is used when method is not directly associated to object lifecycle. Also, the observer lives longer, and it can be detached or attached at any time. For example, displaying values from a model in the UI and updating model from user input.
* **Rails Callback:** Callbacks are methods, which can be called at certain moments of an object’s life cycle for example it can be called when an object is validated, created, updated, deleted, A call back is short lived. For example, running a thread and giving a call-back that is called when thread terminates

**25) Explain what is rake in Rails?**

Rake is a Ruby Make; it is a Ruby utility that substitutes the Unix utility ‘make’, and uses a ‘Rakefile’ and ‘.rake files’ to build up a list of tasks. In Rails, Rake is used for normal administration tasks like migrating the database through scripts, loading a schema into the database, etc.

**26) Explain how you can list all routes for an application?**

To list out all routes for an application you can write rake routes in the terminal.

**27) Explain what is sweeper in Rails?**

Sweepers are responsible for expiring or terminating caches when model object changes.

**28) Mention the log that has to be seen to report errors in Ruby Rails?**

Rails will report errors from Apache in the log/Apache.log and errors from the Ruby code in log/development.log.

**29) Explain what is the difference between Dynamic and Static Scaffolding?**

|  |  |
| --- | --- |
| **Dynamic Scaffolding** | **Static Scaffolding** |
| * It automatically creates the entire content and user interface at runtime * It enables to generation of new, delete, edit methods for the use in application * It does not need a database to be synchronized | * It requires manual entry in the command to create the data with their fields * It does not require any such generation to take place * It requires the database to be migrated |

**30) Mention what is the function of garbage collection in Ruby on Rails?**

The functions of garbage collection in Ruby on Rails includes

* It enables the removal of the pointer values which is left behind when the execution of the program ends
* It frees the programmer from tracking the object that is being created dynamically on runtime
* It gives the advantage of removing the inaccessible objects from the memory, and allows other processes to use the memory

**31) Mention what is the difference between redirect and render in Ruby on Rails?**

* Redirect is a method that is used to issue the error message in case the page is not issued or found to the browser. It tells browser to process and issue a new request.
* Render is a method used to make the content. Render only works when the controller is being set up properly with the variables that require to be rendered.

**32) Mention what is the purpose of RJs in Rails?**

RJs is a template that produces JavaScript which is run in an eval block by the browser in response to an AJAX request. It is sometimes used to define the JavaScript, Prototype and helpers provided by Rails.

**33) Explain what is Polymorphic Association in Ruby on Rails?**

Polymorphic Association allows an ActiveRecord object to be connected with Multiple ActiveRecord objects. A perfect example of Polymorphic Association is a social site where users can comment on anywhere whether it is a videos, photos, link, status updates etc. It would be not feasible if you have to create an individual comment like photos\_comments, videos\_comment and so on.

**34) Mention what are the limits of Ruby on Rails?**

Ruby on Rails has been designed for creating a CRUD web application using MVC. This might make Rails not useful for other programmers. Some of the features that Rails does not support include

* Foreign key in databases
* Linking to multiple data-base at once
* Soap web services
* Connection to multiple data-base servers at once

|  |
| --- |
| Ruby on rails interview questions and answers |
| 1. [What does ruby name refer to? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2439)2.[What is the difference between Symbol and String? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2438)3.[What does Session and Cookies represent in Ruby on Rails?](http://www.careerride.com/view.aspx?id=2437)4.[Write a program to show the functionality of request.xhr in Ruby on Rails.](http://www.careerride.com/view.aspx?id=2436)5.[What is the role of MVC architecture in Ruby on Rails?](http://www.careerride.com/view.aspx?id=2435)6.[What are the components defined in the model from MVC architecture? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2434)7.[What is the function of ORM in Ruby on Rails?](http://www.careerride.com/view.aspx?id=2433)8.[What are the different types of association relationships that exist? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2432)9.[What is the difference between render and redirect? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2431)10.[What is the difference between Static and Dynamic Scaffolding? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2430)11.[Write a program to generate and run the application without the use of database - Ruby on Rails](http://www.careerride.com/view.aspx?id=2429)12.[Write a program to show the usage of sql in the database file? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2428)13.[What is the main function of helpers used in Ruby on Rails?](http://www.careerride.com/view.aspx?id=2427)14.[What are the different components of Rails? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2426)15.[What is the directory structure of Rails? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2425)16.[What are the functions performed by Rails migration? - Ruby on Rails](http://www.careerride.com/view.aspx?id=2424)17.[What is the use of $ in Ruby?](http://www.careerride.com/view.aspx?id=2423)18.[What are the steps involved in writing and setting up an application in Ruby on Rails?](http://www.careerride.com/view.aspx?id=2422)19.[What is the function of garbage collection in Ruby on Rails?](http://www.careerride.com/view.aspx?id=2421)20.[What is the purpose of load, auto\_load, and require\_relative in Ruby?](http://www.careerride.com/view.aspx?id=2419)  |  |  | | --- | --- | | |  | | --- | |  | | |

When hiring Ruby on Rails programmers, knowing the right questions to ask during an interview was a real challenge for me at first. In 30 minutes or less, it's difficult to get a solid read on a candidate's skill set without looking at code they've previously written. And in the corporate/enterprise world, I often don't have access to their previous work.

To ensure we hired competent ruby developers at my last [job](http://www.tobi.com/), I created a list of 15 ruby questions -- a ruby measuring stick if you will -- to select the cream of the crop that walked through our doors.

## What to expect

Candidates will typically give you a range of responses based on their experience and personality. So it's up to you to decide the correctness of their answer.

There are **many** solutions to most of these questions -- some aren't listed here. Candidates get my respect for knowing esoteric solutions, but I'm looking for developers with similar practices to my own.

Make no mistake, this list is **not complete**, by any means. But it does provide a useful tool to put a candidate's reaction to the same series of questions in perspective. Especially during an interview, when your mouth goes dry and and your hands get all sweaty.

## Begin!

Senior programmers won't have a problem with these, while junior programmers will usually give only half-answers.

#### What is a class?

A text-book answer: classes are a blue-print for constructing computer models for real or virtual objects... boring.

In reality: classes hold **data**, have **methods** that interact with that data, and are used to **instantiate objects**.

Like this.

class WhatAreClasses

def initialize

@data = "I'm instance data of this object. Hello."

end

def method

puts @data.gsub("instance", "altered")

end

end

object = WhatAreClasses.new

object.method

#=> I'm altered data of this object. Hello.

#### What is an object?

An instance of a class.

To some, it's also the root class in ruby (Object).

Classes themselves descend from the Object root class. (Kudos to Ezra)

#### What is a module? Can you tell me the difference between classes and modules?

Modules serve as a mechanism for **namespaces**.

module ANamespace

class AClass

def initialize

puts "Another object, coming right up!"

end

end

end

ANamespace::AClass.new

#=> Another object, coming right up!

Also, modules provide as a mechanism for multiple inheritance via **mix-ins** and **cannot be instantiated** like classes can.

module AMixIn

def who\_am\_i?

puts "An existentialist, that's who."

end

end

# String is already the parent class

class DeepString < String

# extend adds instance methods from AMixIn as class methods

extend AMixIn

end

DeepString.who\_am\_i?

#=> An existentialist, that's who.

AMixIn.new

#=> NoMethodError: undefined method ‘new’ for AMixIn:Module

#### Can you tell me the three levels of method access control for classes and modules? What do they imply about the method?

**All methods**, no matter the access control, can be **accessed within the class**. But what about outside callers?

Public methods enforce **no access control** -- they can be called in any scope.

Protected methods are only accessible to **other objects of the same class**.

Private methods are only accessible within the **context of the current object**.

class AccessLevel

def something\_interesting

another = AccessLevel.new

another.public\_method

another.protected\_method

another.private\_method

end

def public\_method

puts "Public method. Nice to meet you."

end

protected

def protected\_method

puts "Protected method. Sweet!"

end

private

def private\_method

puts "Incoming exception!"

end

end

AccessLevel.new.something\_interesting

#=> Public method. Nice to meet you.

#=> Protected method. Sweet!

#=> NoMethodError: private method ‘private\_method’ called for

#=> #<AccessLevel:0x898c8>

#### There are three ways to invoke a method in ruby. Can you give me at least two?

Here, I'm looking for the **dot operator** (or period operator), the **Object#send** method, or **method(:foo).call**

object = Object.new

puts object.object\_id

#=> 282660

puts object.send(:object\_id)

#=> 282660

puts object.method(:object\_id).call # (Kudos to Ezra)

#=> 282660

## Separating the professional from the hobbyist

Senior programmers should be able to give competent answers for all questions. Junior programmers should answer some correct, but usually won't know them all.

#### Explain this ruby idiom: a ||= b

A common idiom that strong ruby developers use all the time.

# a = b when a == false

# otherwise a remains unchanged

a || a = b # (Kudos to Markus Prinz)

a = 1

b = 2

a ||= b #=> a = 1

a = nil

b = 2

a ||= b #=> a = 2

a = false

b = 2

a ||= b #=> a = 2

#### What does self mean?

self always refers to the current object. But this question is more difficult than it seems because **Classes are also objects** in ruby. (Kudos to Stephen)

class WhatIsSelf

def test

puts "At the instance level, self is #{self}"

end

def self.test

puts "At the class level, self is #{self}"

end

end

WhatIsSelf.test

#=> At the class level, self is WhatIsSelf

WhatIsSelf.new.test

#=> At the instance level, self is #<WhatIsSelf:0x28190>

This short snippet indicates two things:

* at the class level, self is the **class**, in this case WhatIsSelf.
* at the instance level, self is the **instance in context**, in this case the instance of WhatIsSelf at memory location 0x28190.

#### What is a Proc?

Everyone usually confuses procs with blocks, but the strongest rubyist can grok the true meaning of the question.

Essentially, Procs are **anonymous methods** (or nameless functions) containing code. They can be placed inside a variable and **passed around** like any other object or scalar value. They are created by **Proc.new**, **lambda**, and **blocks** (invoked by the yield keyword).

**Note:** Procs and lambdas do have subtle, but important, [differences](http://en.wikibooks.org/wiki/Ruby_Programming/Syntax/Method_Calls#Understanding_blocks.2C_Procs_and_methods) in ruby v1.8.6. However, I wouldn't expect a candidate talk about these nitty-gritty details during an interview. (Kudos to Noah Thorp)

# wants a proc, a lambda, AND a block

def three\_ways(proc, lambda, &block)

proc.call

lambda.call

yield # like block.call

puts "#{proc.inspect} #{lambda.inspect} #{block.inspect}"

end

anonymous = Proc.new { puts "I'm a Proc for sure." }

nameless = lambda { puts "But what about me?" }

three\_ways(anonymous, nameless) do

puts "I'm a block, but could it be???"

end

#=> I'm a Proc for sure.

#=> But what about me?

#=> I'm a block, but could it be???

#=> #<Proc:0x00089d64> #<Proc:0x00089c74> #<Proc:0x00089b34>

#### What is unit testing (in classical terms)? What is the primary technique when writing a test?

The strongest candidates should be quite comfortable with test or behavior driven development.

Unit testing, simply put, is testing methods -- the smallest unit in object-oriented programming. Strong candidates will argue that it allows a developer to flesh out their API before it's consumed by other systems in the application.

The primary way to achieve this is to **assert** that the **actual** result of the method matches an **expected** result.

require "test/unit"

class Brokened

def uh\_oh

"I needs fixing"

end

end

class BrokenedTest < Test::Unit::TestCase

def test\_uh\_oh

actual = Brokened.new

assert\_equal("I'm all better!", actual.uh\_oh)

end

end

#=> Started

#=> F

#=> Finished in 0.663831 seconds.

#=>

#=> 1) Failure:

#=> test\_uh\_oh:11

#=> <"I'm all better!"> expected but was

#=> <"I needs fixing">.

#=>

#=> 1 tests, 1 assertions, 1 failures, 0 errors

## Show me the money!

Variable typing is one of those topics that everyone sort of understands it, but is hard to put it into words. I've iterated and improved the next series of questions to really test a **senior level** candidate's knowledge of static and dynamic typing. This is my best attempt so far.

#### What is the primary difference in these two code snippets?

// Java

public boolean isEmpty(String s) {

return s.length() == 0;

}

# ruby

def empty?(s)

return s.size == 0

end

The Java method **only** accepts Strings as arguments and **only** returns a boolean while...

The ruby method accepts **any Object** and **could return anything**, but in this case **will return a boolean** if executed without exceptions.

#### What does this say about the advantages of ruby's dynamic (duck) typed system?

That ruby program use **less code** and are **more flexible**.

#### What are some disadvantages (real and potential)?

Developers cannot be 100% certain that all arguments sent this empty? method will have a size method that is publicly accessible. Also, ruby is an interpreted language and it may take longer to run than compiled programs, such as Java, that are programmed similarly.

#### What could a developer do to address these disadvantages?

She could write unit tests or specs to ensure her application behaves as intended. She could also profile her application with tools like the unix [time](http://unixhelp.ed.ac.uk/CGI/man-cgi?time) command, the ruby [Benchmark](http://www.ruby-doc.org/core/classes/Benchmark.html) class, and the ruby library called [ruby-prof](http://ruby-prof.rubyforge.org/).

A cunning programmer would also argue that these two techniques ought to be used for both static and dynamic languages when developing complex systems.

## Wrapping things up

To finish up with, I like to lob in some easy ones again. Plus I like to scratch my own curiosity about a candidates relationship with the ruby community.

#### What are rubygems? Any favorites not including rails? Any that you've worked on personally?

[rubygems](http://www.rubygems.org/) is package manager software for ruby libraries (i.e. gems). The package manager has basic [CRUD](http://en.wikipedia.org/wiki/Create,_read,_update_and_delete) operations, dependency trees, and supports asynchronous communication between multiple gem servers.

#### What is your favorite api resource for ruby?

I really like [gotAPI](http://www.gotapi.com/rubyrails) -- auto-complete searching for both ruby and rails together!