FullStack.Cafe - Kill Your Tech Interview

Q1: Mention what is Rails *Migration*? ☆

Topics: Ruby on Rails

Answer:

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.

Q2: Explain what is *rake* in Rails? ☆

Topics: Ruby on Rails

Answer:

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.

Q3: What Is ORM In Rails? ☆

Topics: Ruby on Rails

Answer:

ORM tends for Object-Relationship-Model, where Classes are mapped to table in the database, and Objects are directly mapped to the rows in the table.

Q4: Mention what are the positive aspects of Rails? ☆☆

Topics: Ruby on Rails

Answer:

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, development, and production.
- **Built-in-testing**: It supports code called harness and fixtures that make test cases to write and execute.

Q5: Explain what is the role of *sub-directory app/controllers* and *app/helpers*? ☆☆

Topics: Ruby on Rails

Answer:

- 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.

Q6: Explain what is Rails *Active Record* in Ruby on Rails? ☆☆

Topics: Ruby on Rails

Answer:

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

Q7: List out what can Rails *Migration* do? ☆☆

Topics: Ruby on Rails

Answer:

Rails Migration can do following things

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

Q8: Mention what is the role of Rails *Controller*? ☆☆

Topics: Ruby on Rails

Answer:

The Rails **controller** is the logical center of the application. It facilitates 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

Q9: Explain how you define Instance *Variable*, *Global Variable* and *Class Variable* in Ruby? ☆☆

Topics: Ruby on Rails

Answer:

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

Q10: Mention what are the limits of Ruby on Rails? ☆☆

Topics: Ruby on Rails

Answer:

Ruby on Rails has been designed for creating a CRUD web application using MVC. Some of the features that Rails does not support include:

- Foreign key in databases
- Linking to multiple database at once
- Soap web services
- Connection to multiple database servers at once

Q11: Explain what is a *class library* in Ruby? ☆☆

Topics: Ruby on Rails

Answer:

Ruby class libraries consist of a variety of domains, such as thread programming, data types, various domains, etc. These classes give flexible capabilities at a high level of abstraction, giving you the ability to create powerful Ruby scripts useful in a variety of problem domains. The following domains which have relevant class libraries are,

- GUI programming
- Network programming
- CGI Programming
- Text processing

Q12: Mention what is the difference between a *gem* and a *plugin* in Ruby? ☆☆

Topics: Ruby on Rails

Answer:

- **Gem**: A gem is a just ruby code. It is installed on a machine, and it’s available for all ruby applications running on that machine.

- **Plugin**: Plugin is also ruby code, but it is installed in the application folder and only available for that specific application.

Q13: What is the use of `load` and `require` in Ruby? ☆☆

Topics: Ruby on Rails

Answer:

- You use `load()` to execute code
- use `require()` to import libraries.

Q14: What Are The Various *Components* Of Rail? ☆☆

Topics: Ruby on Rails

Answer:

1. **Action Pack**: Action Pack is a single gem that contains Action Controller, Action View and Action Dispatch. The "VC" part of "MVC".

- Action Controller: Action Controller is the component that manages the controllers in a Rails application. The Action Controller framework processes incoming requests to a Rails application, extracts parameters, and dispatches them to the intended action.

- Action View: Action View manages the views of your Rails application. It can create both HTML and XML output by default. Action View manages rendering templates, including nested and partial templates, and includes built-in AJAX support.

- Action Dispatch: Action Dispatch handles routing of web requests and dispatches them as you want, either to your application or any other Rack application. Rack applications are a more advanced topic and are covered in a separate guide called Rails on Rack.

2. **Action Mailer**: Action Mailer is a framework for building e-mail services. You can use Action Mailer to receive and process incoming email and send simple plain text or complex multipart emails based on flexible templates.

3. **Active Model**: Active Model provides a defined interface between the Action Pack gem services and Object Relationship Mapping gems such as Active Record. Active Model allows Rails to utilize other ORM frameworks in place of Active Record if your application needs this.

4. **Active Record**: Active Record are like Object Relational Mapping (ORM), where classes are mapped to table, objects are mapped to columns and object attributes are mapped to data in the table.

5. **Active Resource**: Active Resource provides a framework for managing the connection between business objects and RESTful web services. It implements a way to map web-based resources to local objects with CRUD semantics.

6. **Active Support**: Active Support is an extensive collection of utility classes and standard Ruby library extensions that are used in Rails, both by the core code and by your applications.

Q15: What Do You Mean By `Render` And `Redirect_to`? ☆☆

Topics: Ruby on Rails

Answer:

- *render* causes rails to generate a response whose content is provided by rendering one of your templates. Means, it will direct goes to view page.

- *redirect_to* generates a response that, instead of delivering content to the browser, just tells it to request another url. Means it first checks actions in controller and then goes to view page.

Q16: How Many *Types Of Associations Relationships* Does A Model Have? ☆☆

Topics: Ruby on Rails

Answer:

When you have more than one model in your rails application, you would need to create connection between those models. You can do this via associations. Active Record supports three types of associations:

- **one-to-one**: A one-to-one relationship exists when one item has exactly one of another item. For example, a person has exactly one birthday or a dog has exactly one owner.

- **one-to-many**: A one-to-many relationship exists when a single object can be a member of many other objects. For instance, one subject can have many books.

- **many-to-many**: A many-to-many relationship exists when the first object is related to one or more of a second object, and the second object is related to one or many of the first object.

You indicate these associations by adding declarations to your models:

- `has_one`,
- `has_many`,
- `belongs_to`,
- `has_and_belongs_to_many`

Q17: What Are *Helpers* And How To Use Helpers In ROR? ☆☆

Topics: Ruby on Rails

Answer:

Helpers are modules that provide methods which are automatically usable in your view. They provide shortcuts to commonly used display code and a way for you to keep the programming out of your views. The purpose of a helper is to simplify the view.

Q18: What Is The Difference Between `Nil` And `False` In Ruby? ☆☆

Topics: Ruby on Rails

Answer:

False is a boolean datatype, Nil is not a data type it have object_id 4.

Q19: How Is *Visibility Of Methods* Changed In Ruby (encapsulation)? ☆☆

Topics: Ruby on Rails

Answer:

By applying the access modifier:

- Public,
- Private and
- Protected

Q20: What do you mean by the term *Scaffolding* and what sort of advantages the Ruby can offer when it comes to same? ☆☆

Topics: Ruby on Rails

Answer:

While developing the projects, the users often have to write codes in the early stage of development. These codes help building the application in a very reliable manner and quickly and also, a close eye can be kept on the working of some major components with this approach. In Ruby, the scaffolding is done automatically and the users are free to concentrate on the core development only from the first day of development.