Ruby on rails interview Questions Cheat Sheet

by Akshaya (akki_1) via cheatography.com/208452/cs/44671/

What is Ruby on rails?

It is a Open-source web application framework written in ruby

* It is designed to make web applications easier by assuming what developers need to get started.

* It follows the model-view-controller (MVC) architecture pattern.

MVC Architecture

In Ruby, MVC (Model-View-Controller) is a pattern used to organize code, especially in web applications. It separates the applic-ation into three parts:

Model: Handles data and business logic. For example, it communicates with the database.

View: Manages the display. It's what users see on the screen, like HTML or templates.

Controller: Connects Model and View. It receives user input, tells the Model to update or retrieve data, and then decides what View to show.

Flow: User -> Controller -> Model -> Controller -> View -> User.

In Ruby on Rails, this pattern helps keep code clean and organized, making it easier to manage large applications.

MVC mean

MVC stands for Model-View-Controller. It's a software architectural pattern commonly used in the development of user interfaces, particularly for web applications.

Migration

It is a feature in rails that allow developers to manage & modify database schema over time.

* They are Ruby files used to make changes to the database, like creating tables or adding/removing columns.

* Migrations make it easier to track and version change, rollback changes if necessary

Why Migrations are important

Migrations are important because they help track changes, keep the database schema consistent, and allow rolling back changes if needed.

Rails controller & purpose

Rails **Controller** handles requests and sends responses. It processes user actions, interacts with models to fetch or save data, and renders views or returns data (e.g., JSON).

* Controllers are mapped to URLs using routes. They act as the middle layer between the user and the app's data or views.

Ex: User visits /articles. router directs request to ArticlesController#index, controller feteches all articles with Articl-e.all, controller sends data to view /returns a response.

Active Record

Active Record is the ORM (Object-Rela-tional Mapping) in Rails. It connects Ruby classes to database tables, making it easy to work with the database without writing SQL. Developers can use Ruby methods to interact with the data.

How active record works in database?

Maps models to database tables (e.g., User →

Provides methods for CRUD operations create update, destroy

Ex: create(User.**create**(name: " "), find(User.**fir** update(user.**update**(name: " "), destroy(user.**d**

Uses migrations to manage the database sche

Handles relationships with associations (e.g., belongs_to).

Allows queries using Ruby methods instead of

```
.w her e("age > ?", 18).or der (:n
```

ORM(Object-Relational-Mapping)

It allows developers to interact with database using ruby code instead of writing raw SQL queries.

* In Rails, the ORM is Active Record, which maps models to database tables and provides methods for performing CRUD operations (Create, Read, Update, Delete).

Gemfile

It is used to manage the dependencies of a Rails application. It specifies which gems (libraries) the application depends on and their versions.

Some Gem files

Authentication & Authorization: : Devise

* It handles user authentication(login, signup, password recover)

Database & ORM: bcrypt

* Handles password encryption when using has_secure_password

Frontend: Bootstrap

* Add bootstrap style for responsive web design.

Kamineri: Adds pagination to your applic-ation.

Routing

Routing connects incoming requests (URLs) to the specific controller actions. Rails uses a routes.rb file to define routes.

A route can be defined as: get 'articles', to: 'articles#index'

- a GET request to /articles will be handled by the index action in the ArticlesController.

- Rails has RESTful routes, which map standard URL patterns to controller actions (e.g., index, show, create, etc.) based on resources:

resources :articles

- This generates routes for all CRUD actions (index, show, new, edit, create, update, destroy).

Routes ensure incoming requests are directed to the right controllers and actions, organizing the application's structure effectively.

RESTful Architecture

RESTful architecture organizes web applications around resources and standardizes how they are accessed using HTTP verbs.

Resources: Treats entities (e.g., articles, users) as resources.

HTTP Verbs: Maps actions to HTTP methods:

GET: Retrieve data. **POST**: Create data. **PUT/PATCH**: Update data. **DELETE**: Remove data.

RESTful Routes in Rails: Using resources, Rails generates routes for CRUD operations.

resources :articles

GET /articles → ArticlesController#index (List all articles)

POST /articles → ArticlesController#create (Create a new article).

GET /articles/:id → ArticlesController#show (Show a specific article).

RESTful Architecture (cont)

PATCH/PUT /articles/:id → ArticlesController#update (Update an article).

DELETE /articles/:id → ArticlesController#destroy (Delete an article).

Associations & types

Associations define relationships between models. They allow models to connect and share data easily.

Types

has_many: Defines one-to-many relationship. A model "has many" other models.
Ex: User can have many Posts

belongs_to: Defines the opposite of has_many. A model "belongs to" another model.
Ex: Post belongs to User

has_one: Defines a one-to-one relationship. A model "has one" other model. {{nl}Ex: User has one Profile

has_and_belongs_to_many:Defines a many-to-many relationship without a join model.
Ex: Student can have many Courses, and a Course can have many Students.

has_many :through: Defines a many-to-many relationship with a join model.
Ex: Doctor has many Patients through Appointments.

Polymorphic association

1 table associated with many tables

Ex: Post have likes, we use likes in post and comment

Active Record Vs Active Model

Active Record is an ORM framework in Rails that connects classes to database tables, enabling database operations through Ruby objects

Active Model provides modules for building custom models without a database, offering features like validations, serialization, and callbacks.

How do you handle errors?

Error handling in Rails is important for providing a smooth user experience and ensuring that users are properly informed when something goes wrong.

Partials

Partials in Rails are reusable pieces of view code that can be shared across multiple templates. They help reduce code duplication and improve maintainability by breaking views into smaller, modular components.

* Partial files start with an underscore (_) in their filename, e.g., _header.html.erb. They are called without the underscore, e.g., <%= render 'header' %>.

Sessions

Sessions store user-specific data across requests, typically used for authentication and tracking user activity.

In Rails, session data is stored in a cookie (by default) and is accessible using the session hash.

Use of Rails console

The Rails console is an interactive command-line tool that allows developers to interact with their Rails application directly

Interacting with the Database: Test queries and CRUD operations without writing a full script.

Use of Rails console (cont)

Testing Code: Run snippets of Ruby or Rails code to verify functionality.

Debugging: Inspect data, test methods, or debug application logic.

dependent destroy

Used in associations specifies when record is deleted, all records are also deleted.

* Used in one-many or many-many

Ex: Post have comments, if post deleted comment also deleted

Callbacks

Callbacks are methods that get triggered at certain points in the lifecycle of an object (like before saving, after updating, etc.).

Common callbacks include:
before_save: Runs before a record is saved (on both create and update). Often used for tasks like formatting data.

after_save: Runs after a record is saved. Useful for tasks like logging or sending notifications.

before_validation: Runs before validation. Can be used to modify attributes before validation checks.

after_create: Runs after a new record is created., not on updates. Often used to trigger actions like sending welcome emails.

They are used to execute code at specific points to maintain data consistency or add additional logic.

render vs redirect_to

render: It view template without making a new request(redirecting to browser). Simply display view..

redirect_to: Redirect browser to another action/ URL, triggering new request.

find vs find_by vs where

Find: It looks up a record using its primary key(id). It gives error if records isn't found.

find_by: Searches for first record that matches certain condition, return nil if no match found.

where: Finds all records that meet specific condition & returns them as list of objects.

resource vs resources

resource: It will give routes to all command operations except Index.

resources: It will give routes to all command operations

validate vs validates

validate : It is used to define custom validations method.

validates : It is used to apply built-in validation rules to model attributes.

delete vs destroy

delete: Doesnot execute callbacks.

destroy: executes callbacks

Purpose of rails db:seed task

The rails db:seed task is used to populate the database with initial data.

MVC In general

MVC is a design pattern used across different programming languages, not just in Ruby. It divides code into Model, View, and Controller layers, which helps organize code by separating data handling, display, and logic, making it easier to develop and maintain applications.

Active Record in rails

Active Record is part of the M in MVC pattern — which is the layer of the system responsible for representing data and business logic.

By **Akshaya** (akki_1)
cheatography.com/akki-1/

Published 17th November, 2024.
Last updated 17th November, 2024.