

**WEB TECHNOLOGY UNIT – 6 PYQ'S****➤ MAY / JUN 2022****Q7)****a) Explain classes and objects in Ruby with appropriate examples. [9 Marks]****✓ 1. What is a Class in Ruby?**

A **class** in Ruby is a **blueprint** or **template** used to create objects. It defines properties (using variables) and behaviors (using methods).

```
ruby                                                                    Copy Edit

class Student
  def initialize(name, roll_no)
    @name = name
    @roll_no = roll_no
  end

  def display_details
    puts "Name: #{@name}, Roll No: #{@roll_no}"
  end
end
```

- `initialize` is a **constructor** method.
- `@name` and `@roll_no` are **instance variables**.

**✓ 2. What is an Object in Ruby?**

An **object** is an **instance** of a class. It is created using the `.new` keyword and can access the class's methods and variables.

```
ruby                                                                    Copy Edit

s1 = Student.new("Himanshu", 101)
s1.display_details
```

**Output:**

```
yaml                                                                    Copy Edit

Name: Himanshu, Roll No: 101
```

### 3. Features of Classes & Objects in Ruby

Feature	Description
<b>initialize</b>	Constructor for assigning values
<b>@variable</b>	Instance variable for storing object data
<b>self</b>	Refers to the current object
<b>attr_accessor</b>	Creates getter and setter methods

### 5. Benefits of Classes & Objects

- Code reuse through object instantiation.
- Encapsulation of data and logic.
- Easier to manage complex programs.

In Ruby, classes define structure, and objects bring them to life. They form the basis of object-oriented programming, helping build clean, modular, and reusable code.

### Q7

**b) Introduce the concept of Rails application. Describe layouts & stylesheet in Rails. [8 Marks]**

#### 1. Introduction to Rails Application

Ruby on Rails (or Rails) is a full-stack web development framework written in Ruby. It follows the MVC (Model-View-Controller) architecture and emphasizes:

- Convention over Configuration
- DRY (Don't Repeat Yourself)
- Rapid Development

**A Rails application provides tools to handle everything from:**

- Database operations (Model),
- Business logic (Controller),
- Web page rendering (View).

**Command to create a Rails app:**

```
bash

rails new blog_app
```

[Copy](#) [Edit](#)**2. Layouts in Rails**

Layouts in Rails are templates that wrap around views to provide a common structure (like headers, footers, and menus).

- Stored in: `app/views/layouts/application.html.erb`
- All views are rendered inside the `<%= yield %>` tag.

**Example Layout:**

```
erb

<!DOCTYPE html>
<html>
<head>
  <title>MyApp</title>
  <%= stylesheet_link_tag 'application', media: 'all' %>
</head>
<body>
  <%= render 'shared/navbar' %>
  <%= yield %>
  <%= render 'shared/footer' %>
</body>
</html>
```

[Copy](#) [Edit](#)

- `yield` : Placeholder where specific view content is inserted.
- `render` : Used to include partials like header or footer.

**3. Stylesheets in Rails**

Rails uses CSS (or SCSS) files to style HTML content. These stylesheets are stored in:

```
bash

app/assets/stylesheets/
```

[Copy](#) [Edit](#)

**Key points:**

- By default, Rails includes application.css or application.scss.
- You can create custom stylesheets per controller or component.

**Adding a CSS class:**

```
erb                                                                    Copy Edit

<h1 class="title">Welcome</h1>
```

**Corresponding CSS (in application.css):**

```
css                                                                    Copy Edit

.title {
  color: green;
  font-size: 24px;
}
```

A Rails application is a powerful structure for building web apps. Layouts ensure a consistent structure across pages, and stylesheets provide design and aesthetics, all while following best coding practices.

**Q8****a) Explain scalar types and their operations in Ruby. [9 Marks]**

Scalar types in Ruby represent **single, simple values** (not collections like arrays or hashes). Ruby treats everything as an object, even scalar types, and provides methods for various operations.

**Common Scalar Types in Ruby :**

Type	Description	Example
Integer	Whole numbers	10, -5, 0
Float	Decimal numbers	3.14, -0.5
String	Sequence of characters	"Ruby", 'Tech'
Symbol	Lightweight string-like immutable values	:name, :email

<b>Boolean</b>	Truth values	true, false
<b>NilClass</b>	Represents 'nothing' or 'null'	nil

### Operations on Scalar Types :

### A) Integer and Float Operations

- **Arithmetic:** `+`, `-`, `*`, `/`, `%`, `**`

```
ruby
```

Copy

Edit

```
a = 10
b = 3.5
puts a + b      # 13.5
puts a % 3      # 1
```

- **Comparison:** `==`, `!=`, `<`, `>`, `<=`, `>=`

```
puts a > b      # true
```

## B) String Operations

- Concatenation: `+` or `<<`
- Repetition: `*`
- Interpolation: `"Hello #{name}"`

```
ruby

str1 = "Hello"
str2 = "World"
puts str1 + " " + str2      # Hello World
puts "Ruby" * 3             # RubyRubyRuby
```

 Copy  Edit

## C) Symbol Operations

- Used for identifiers, keys in hashes, etc.
- Symbols are immutable and memory-efficient.

```
ruby

:name == :name      # true
```

 Copy  Edit

#### D) Boolean Operations

- `&&` (AND), `||` (OR), `!` (NOT)

ruby

Copy

Edit

```
x = true
y = false
puts x && y      # false
puts !x          # false
```

#### E) NilClass Operations

- `nil` represents no value. Returns `false` in boolean context.

ruby

Copy

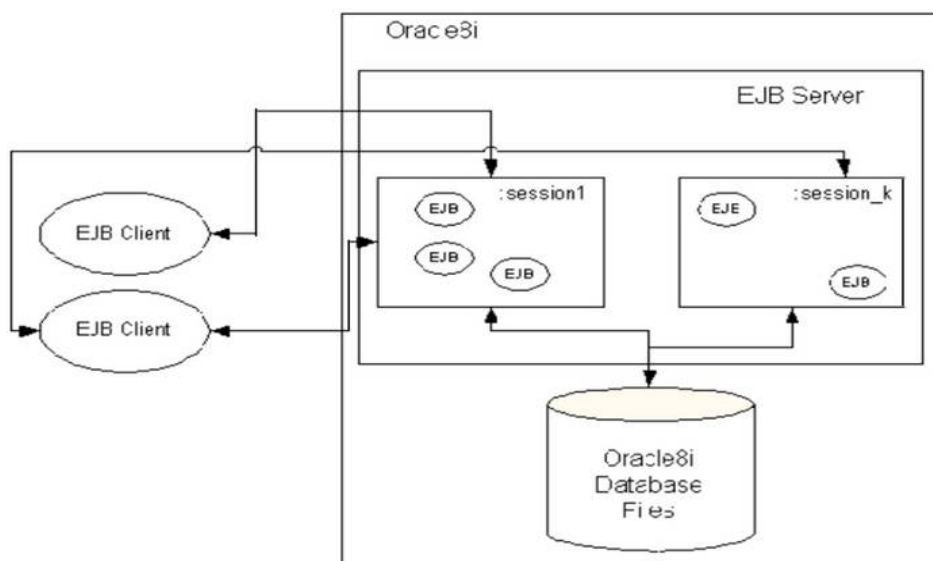
Edit

```
puts nil.nil?    # true
puts !nil        # true
```

Ruby's scalar types are object-oriented, flexible, and support rich operations. These types are essential for control flow, data processing, and method return values.

Q8

b) Explain Architecture of EJB & explain types of EJB in detail. [9 Marks]



**Enterprise JavaBeans (EJB)** is a server-side component architecture for developing **modular**, **scalable**, and **secure** enterprise applications in Java. It provides built-in support for **transactions**, **security**, **concurrency**, and **persistence**.

## 2. Components of EJB Architecture :

### EJB Component (Bean Class)

- Contains the actual **business logic**.
- Types: Session Beans, Message-Driven Beans (MDB).

### EJB Container

- Manages bean lifecycle, transactions, security, threading, and pooling.
- Automatically provides system-level services.

### Java EE Server (Application Server)

- Hosts the EJB container (e.g., GlassFish, WildFly).
- Coordinates with database and clients.

### Client

- Can be web apps, desktop apps, or other enterprise beans.
- Communicates via Remote/Local Interfaces.

### Database

- Stores the persistent data.
- Accessed via Entity Beans (deprecated) or JPA.

## 3. Types of EJBs :

### Session Beans – Handle business logic

- **Stateless** – No client state across calls.
- **Stateful** – Maintains client state across session.
- **Singleton** – Single shared instance.

### Entity Beans (Deprecated)

- Used to map Java objects to database records.
- Replaced by JPA.

### Message-Driven Beans (MDB)

- Used for asynchronous communication via JMS.



- Doesn't return a response to the sender.

➤ **NOV / DEC 2022**

**Q7**

a) Explain scalar types and their operations in Ruby. [9 Marks]

➔ Already covered

b) What are the positive aspects of Rails, explain with example. [9 Marks]

Ruby on Rails (RoR) is a server-side web application framework written in Ruby. It follows the Model-View-Controller (MVC) architecture and emphasizes convention over configuration and DRY (Don't Repeat Yourself) principles.

**Positive Aspects of Rails :**

◆ 1) **Convention over Configuration**

- Reduces the need for configuration files.
- Rails assumes sensible defaults (e.g., table names, file paths).

**Example:**

ruby

Copy

Edit

```
class Product < ApplicationRecord
end
```

Automatically maps to `products` table in the database without extra configuration.

## ◆ 2) Built-in MVC Architecture

- Separates business logic, UI, and data.
- Easier maintenance and scaling.

### Structure:

- **Model:** Handles data ( `Product.rb` )
  - **View:** Handles UI ( `index.html.erb` )
  - **Controller:** Handles logic ( `products_controller.rb` )
- 

## ◆ 3) Rapid Development

- Scaffolding generates basic app structure automatically.

```
bash

rails generate scaffold Product name:string price:decimal
```

Copy Edit

Creates model, view, controller, migration, and routes instantly.

## ◆ 4) Database Integration with ActiveRecord

- Simplifies database operations using object-oriented syntax.

```
ruby

Product.find(1)
Product.create(name: "Mango", price: 50)
```

Copy Edit

## ◆ 5) RESTful Design

- Rails encourages RESTful routing for clean and meaningful URLs.

```
ruby

# routes.rb
resources :products
```

Copy Edit

Creates routes like `/products`, `/products/:id/edit`, etc.

#### ◆ 6) Security Features

- Built-in protection against **SQL Injection**, **Cross-Site Scripting (XSS)**, and **CSRF**.

#### ◆ 7) AJAX Integration

- Rails supports AJAX for dynamic page updates without reload.

erb

Copy

Edit

```
<%= link_to "Delete", product_path(product), method: :delete, remote: true %>
```

Rails promotes **rapid, secure, and maintainable** web development with its rich features and philosophy. It is ideal for startups and projects needing quick turnaround with scalability.

#### Q8 a) Write short note on: [9 Marks]

##### i) Rails with AJAX

##### ii) WAP and WML

##### i) Rails with AJAX

AJAX (**Asynchronous JavaScript and XML**) allows web pages to update parts of the page without reloading the entire page. Ruby on Rails integrates AJAX seamlessly using **remote links/forms** and **UJS (Unobtrusive JavaScript)**.

#### Key Features:

- Improves **user experience** by making apps faster and smoother.
- Enables **partial page updates** using .js.erb files.
- Uses remote: true to make AJAX requests.

**EXAMPLE :**

```
erb
Copy Edit

<%= form_with(model: @post, remote: true) do |f| %>
  <%= f.text_field :title %>
  <%= f.submit "Save" %>
<% end %>
```

**Benefits:**

- No full page reload.
- Better performance.
- Enhanced interactivity in Rails apps.

**ii) WAP and WML****WAP (Wireless Application Protocol):**

- A **standard protocol** that enables **mobile devices** (like early cellphones) to access the internet.
- Acts as a bridge between mobile networks and web servers.
- Works on **limited bandwidth** and small screens.

**WML (Wireless Markup Language):**

- A **markup language** designed for WAP.
- Similar to HTML but optimized for mobile devices.
- Uses cards and decks for navigation.

**Example:**

```
xml
Copy Edit

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM/DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1" title="Welcome">
    <p>Welcome to WAP Page</p>
  </card>
</wml>
```

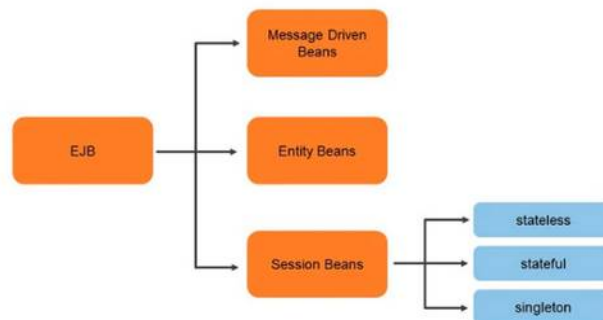
**Key Points:**

- WAP and WML were used before smartphones and responsive web design.
- Now mostly obsolete, replaced by **HTML5** and mobile-optimized websites.

Rails with AJAX enhances interactivity in modern web apps, while WAP and WML laid the foundation for early mobile web browsing.

**Q8**

**b) What is EJB? Explain types of EJBs. [9 Marks]**



**What is EJB? (Enterprise Java Bean)**

Enterprise Java Beans (EJB) is a server-side component architecture used to build scalable, secure, and transactional enterprise applications in Java.

It is part of the **Java EE (Enterprise Edition)** platform.

EJB handles:

- Transactions
- Security
- Multithreading
- Persistence

Developers focus on **business logic**, while the EJB container manages **infrastructure-level concerns**.

**Types of EJBs :**

**1) Session Beans**

Used for business logic that runs in a short-lived session.

Types:

- **Stateless Session Bean**
  - Does not maintain client state between method calls.
  - Example: Currency conversion service.
- **Stateful Session Bean**
  - Maintains state of a conversation with a client.
  - Example: Shopping cart in an e-commerce app.
- **Singleton Session Bean**
  - A single shared instance for all clients.
  - Example: Application-wide cache or configuration service.

## 2) Entity Beans (Deprecated in EJB 3.0)

- Used to represent persistent data stored in a database.
- Each bean instance corresponds to a row in a database.
- Replaced by JPA (Java Persistence API) in modern Java EE.

## 3) Message-Driven Beans (MDB)

- Used for asynchronous messaging.
- Listens to a Java Message Service (JMS) queue or topic.
- Does not return a response to the sender.

### Example Use Case:

Order processing system that handles orders from a JMS queue asynchronously.

## ➤ MAY / JUN 2023

Q7)

a) Explain how multiple selection constructs are implemented in Ruby. [7]

In Ruby, **multiple selection constructs** are implemented using the case statement. It is similar to switch-case in other languages and is used to handle **multiple conditional branches** based on a single expression's value.

### Syntax of `case` Statement:

```
ruby                                                                    Copy Edit

case expression
when value1
  # code block for value1
when value2
  # code block for value2
when value3, value4
  # code block for value3 or value4
else
  # default block if no match
end
```

- **expression:** Evaluated once at the start.
- **when:** Compares using `===` (case equality).
- **else:** Optional default branch.

### Example:

```
ruby                                                                    Copy Edit

grade = "B"

case grade
when "A"
  puts "Excellent"
when "B"
  puts "Good"
when "C"
  puts "Average"
else
  puts "Fail"
end
```

### Key Points:

- case evaluates an expression and compares it using `===` with when values.
- Multiple values can be checked in a single when using commas.
- else acts as a default block when no match is found.

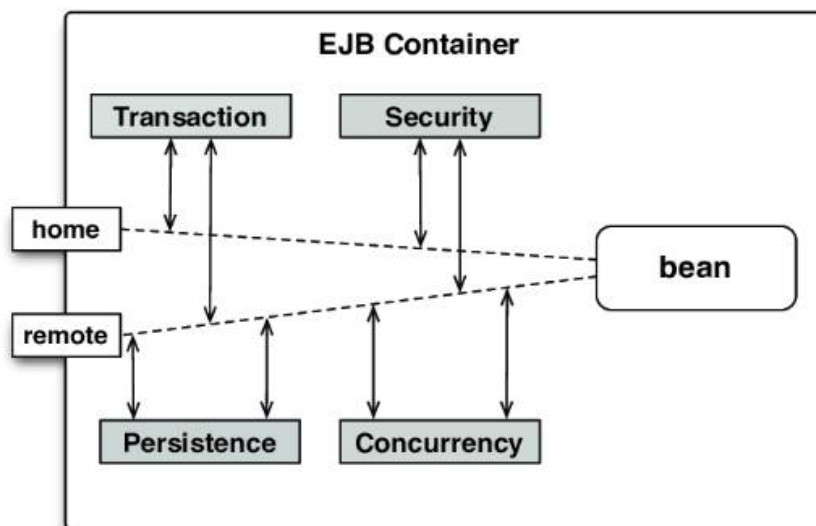
- It improves readability compared to multiple if-elsif-else statements.

The case-when construct in Ruby provides a clean and efficient way to handle multiple conditions based on a single value, making the code easier to read and maintain.

b) Explain Rails with AJAX in detail. [5]

→ Already covered !

c) Draw & explain the role of EJB container in Enterprise applications. [5]



The **EJB Container** is a runtime environment that manages enterprise beans and provides essential system-level services like:

- **Transactions:** Automatically manages commit and rollback to maintain data integrity.
- **Security:** Handles authentication and authorization to secure business logic.
- **Concurrency:** Manages concurrent access ensuring thread safety.
- **Persistence:** Supports entity bean persistence in databases.
- **Lifecycle Management:** Controls creation, pooling, activation, and destruction of beans.

Clients (web, mobile, or Java applications) access EJBs via JNDI lookup or RMI, and the container acts as middleware, abstracting complexities and allowing developers to focus on business logic.

Q8)

a) Explain how to write the methods and call the method in RUBY with example. [7]



## Writing and Calling Methods in Ruby :

### 1. Defining a Method:

- In Ruby, methods are defined using the `def` keyword, followed by the method name and an optional list of parameters.
- The method body contains the code to be executed.
- The method ends with the `end` keyword.
- Methods can return a value explicitly using `return` or implicitly by the last evaluated expression.

#### Syntax:

```
ruby                                                                    Copy Edit

def method_name(parameters)
  # method body
  # optional return
end
```

### 2. Calling a Method:

- To call a method, simply write the method name followed by parentheses enclosing any required arguments.
- Parentheses are optional if no ambiguity arises.

#### Example:

```
ruby                                                                    Copy Edit

# Define a method to greet a person by name
def greet(name)
  return "Hello, #{name}!"
end

# Call the method and store the result
message = greet("Himanshu")

# Print the returned message
puts message
```

**Explanation:**

- The method greet takes one parameter name.
- It returns a greeting string using string interpolation.
- The method is called with the argument "Himanshu".
- The returned string is printed: **Hello, Himanshu!**

In Ruby, methods are defined with def and called by their name with arguments. The last evaluated expression is returned automatically, so explicit return is optional.

**b) What are the difference between java beans and EJB? [5]**

Aspect	JavaBeans	Enterprise JavaBeans (EJB)
<b>Purpose</b>	Reusable software components for GUI and logic encapsulation	Server-side components for distributed, transactional, and secure enterprise applications
<b>Environment</b>	Run on client side or in simple Java apps	Run inside an EJB container on an application server
<b>Services Provided</b>	No built-in services like transactions or security	Provides built-in services like transactions, security, persistence, concurrency
<b>Complexity</b>	Simple and lightweight	More complex, designed for large-scale enterprise systems
<b>Deployment</b>	Can be used anywhere, no special container required	Requires EJB container to manage lifecycle and services

**c) What is Ruby programming language? List some features of Ruby. [5]**

Ruby is a high-level, interpreted, object-oriented programming language designed for simplicity and productivity. It was created by Yukihiro Matsumoto in the mid-1990s. Ruby emphasizes human-friendly syntax and allows developers to write clean and readable code.

**Features of Ruby :**

1. **Pure Object-Oriented:** Everything in Ruby is an object, including numbers, strings, and even classes.
2. **Dynamic Typing:** No need to declare variable types explicitly; types are checked at runtime.
3. **Flexible Syntax:** Supports concise and natural code writing with fewer rules and syntactic sugar.
4. **Garbage Collection:** Automatic memory management to free unused objects.
5. **Rich Standard Library:** Comes with many built-in classes and modules for various tasks.
6. **Exception Handling:** Provides robust error and exception handling mechanisms.
7. **Mixins and Modules:** Supports code reuse and multiple inheritances through modules and mixins.
8. **Duck Typing:** Emphasizes behavior over class type, allowing flexible and adaptable code.

---

➤ **NOV / DEC 2023**

Q7)

a) Explain scalar types, operations and pattern matching in Ruby. [9]

→ already cover + add this

### 3. Pattern Matching in Ruby

Pattern matching was introduced in Ruby 2.7 to simplify complex conditional checks.

- **Purpose:** It matches an expression against a pattern and executes code if it fits.
- **Syntax:** Uses the `case` statement with `in` keyword.

**Example:**

```

ruby                                                                    Copy Edit

case [1, 2, 3]
in [a, b, c]
  puts "Matched with values #{a}, #{b}, and #{c}"
else
  puts "No match"
end
  
```

- Patterns can match arrays, hashes, constants, and use guards (conditions).
- Pattern matching helps destructure data structures and write cleaner code instead of nested if-else.

## b) Explain documents requests and processing forms in Rails. [8]

### 1. Document Requests in Rails

- In Rails, **document requests** refer to HTTP requests sent by clients (browsers) to the Rails server to access resources like web pages, data, or services.
- Rails follows the **MVC (Model-View-Controller)** architecture, where requests are routed to specific controllers based on URL patterns.
- The **Router** (config/routes.rb) maps URLs to controller actions.
- Example: When a user visits `/articles/1`, Rails routes this GET request to the `show` action of the `ArticlesController`.
- Rails supports different HTTP methods:
  - GET (read data),
  - POST (create data),
  - PATCH/PUT (update data),
  - DELETE (delete data).
- Controllers receive the request, interact with models (data), and render views (HTML or JSON) as responses.

## 2. Processing Forms in Rails

- Forms are a common way to collect user input and send data to the server.
- In Rails, **form helpers** simplify creating HTML forms tied to models.
- The main helpers include `form_with`, `form_for` (deprecated), and `form_tag`.

Example of a form to create a new Article:

```
ruby                                                                    Copy Edit

<%= form_with model: @article, local: true do |form| %>
  <%= form.label :title %>
  <%= form.text_field :title %>

  <%= form.label :body %>
  <%= form.text_area :body %>

  <%= form.submit "Create Article" %>
<% end %>
```

- When the form is submitted, it sends data (usually via POST) to the controller action (e.g., `create`).
- In the controller, the submitted parameters are accessed using `params` hash:

Q8)

### a) Explain the concept of classes and arrays in Ruby [9]

#### 1. Classes in Ruby

- A **class** in Ruby is a blueprint for creating objects (instances). It encapsulates data (attributes) and behavior (methods).
- Ruby is fully object-oriented, so everything is an object, and classes are the foundation for defining custom types.
- You define a class using the `class` keyword:

```
ruby                                                                    Copy Edit

class Person
  def initialize(name, age)
    @name = name      # instance variable
    @age = age
  end

  def greet
    puts "Hello, my name is #{@name} and I am #{@age} years old."
  end
end
```

- The initialize method is a special constructor called when creating a new object.
- Objects are created with new:

```
ruby                                                                    Copy Edit

person1 = Person.new("Alice", 30)
person1.greet # Outputs: Hello, my name is Alice and I am 30 years old.
```

## 2. Arrays in Ruby

- An **array** is an ordered, integer-indexed collection that can hold objects of any type (numbers, strings, even other arrays or objects).
- Arrays are flexible and dynamic in size.

Creating arrays:

```
ruby                                                                    Copy Edit

arr = [1, 2, 3, 4]
names = ["Alice", "Bob", "Charlie"]
mixed = [1, "two", 3.0, :four]
```

- Access elements using zero-based indices:

```
ruby                                                                    Copy Edit

puts arr[0]    # 1
puts names[2]  # Charlie
```

Arrays support many useful methods such as:

- `push / <<` — add elements to the end
- `pop` — remove the last element
- `length` or `size` — get the number of elements
- `each` — iterate over elements
- `map` — transform elements

Example:

```
ruby

arr.push(5)
arr.each { |num| puts num * 2 }
```

Copy Edit

- Arrays can be nested and are very versatile in Ruby programming.

## b) Explain concepts of Rails with AJAX and EJB. [8]

### i) Rails with AJAX

AJAX (Asynchronous JavaScript and XML) allows web pages to update parts of the page without reloading the entire page. Ruby on Rails integrates AJAX seamlessly using remote links/forms and UJS (Unobtrusive JavaScript).

#### Key Features:

- Improves user experience by making apps faster and smoother.
- Enables partial page updates using `.js.erb` files.
- Uses `remote: true` to make AJAX requests.

#### EXAMPLE :

```
erb

<%= form_with(model: @post, remote: true) do |f| %>
  <%= f.text_field :title %>
  <%= f.submit "Save" %>
<% end %>
```

Copy Edit

#### Benefits:

- No full page reload.
- Better performance.
- Enhanced interactivity in Rails apps.

## **ii) Rails with EJB**

Ruby on Rails with EJB enables the integration of a lightweight, fast web frontend (Rails) with a robust, scalable enterprise backend (EJB).

This approach is used when existing Java-based enterprise systems (using EJB) need to be accessed or extended using modern Rails applications.

### **Key Features:**

- Separation of concerns: Rails manages the UI and routing, while EJB handles business logic, transactions, and security.
- Communication via APIs: Rails uses RESTful or SOAP web services to call EJB methods remotely.
- Cross-platform integration: Combines the simplicity of Ruby with the scalability of Java EE.
- Reusability: Existing EJB modules can be reused in new Rails-based applications without rewriting them.

### **Benefits:**

- Enables reuse of existing Java enterprise components.
- Suitable for building scalable and hybrid web applications.

## **➤ MAY / JUN 2024**

**Q7)**

**a) Explain Ruby with its advantages. Explain control statements in Ruby.[10]**

### **Ruby Programming Language:**

Ruby is a dynamic, object-oriented, open-source programming language known for its simplicity and productivity. It was developed by Yukihiro “Matz” Matsumoto and blends features from Perl, Smalltalk, and Python.



### Advantages of Ruby:

- Simple and readable syntax – Code looks like natural English.
- Fully object-oriented – Everything is treated as an object.
- Rich libraries (Gems) – Thousands of reusable packages are available.
- Flexible and dynamic – Allows metaprogramming and dynamic typing.

### Control Statements in Ruby:

Control statements manage the flow of execution in a Ruby program. They include conditional and looping structures.

#### 1. Conditional Statements:

- **if, elsif, else:**

```
ruby

age = 18
if age >= 18
  puts "Adult"
else
  puts "Minor"
end
```

Copy Edit

- **unless:**

```
ruby

puts "Access denied" unless logged_in
```

Copy Edit

- **case...when:**

```
ruby

grade = "B"
case grade
when "A"
  puts "Excellent"
when "B"
  puts "Good"
else
  puts "Try harder"
end
```

Copy Edit

**2. Looping Statements:**

Saved memory full ⓘ

- while loop:

```

ruby                                                                    Copy Edit

i = 1
while i <= 5
  puts i
  i += 1
end

```

- for loop:

```

ruby                                                                    Copy Edit

for i in 1..3
  puts i
end

```

- each loop:

```

ruby                                                                    Copy Edit

[1,2,3].each do |num|
  puts num
end

```

- until loop:

```

ruby                                                                    Copy Edit

i = 0
until i > 3
  puts i
  i += 1
end

```

**b) Explain EJB concept & five basic example of using EJB [7]****EJB (Enterprise JavaBeans) Concept:**

**Enterprise JavaBeans (EJB)** is a **server-side component** architecture in Java EE used to build **scalable, secure, and transactional** enterprise applications. EJBs run inside an **EJB container** on an application server and handle business logic.

**Key Features:**

- Manages **transactions, security, persistence, concurrency**
- Simplifies development of **distributed** and **multi-tier applications**
- Supports **declarative programming** through annotations and XML

#### Types of EJB:

- **Session Beans** (Stateless, Stateful, Singleton)
- **Message-Driven Beans (MDBs)**

#### Five Basic Examples of Using EJB:

1. **User Authentication System**  
Use EJB to manage secure login and session tracking.
2. **Online Banking System**  
Use Stateless Beans for fund transfer, balance inquiry, and transaction history.
3. **Shopping Cart in E-commerce**  
Use Stateful Session Bean to maintain a user's cart state.
4. **Email Notification Service**  
Use Message-Driven Bean to process email alerts asynchronously.
5. **Inventory Management System**  
Use EJB for handling stock levels, order processing, and restocking logic.

Q8)

- a) Explain the arrays in Ruby. Explain Rails with AJAX [10]  
➔ Already explain in 2 different question find it and study !!

- b) Explain Document Request in Rails. [4]

#### Document Requests in Rails

- In Rails, **document requests** refer to HTTP requests sent by clients (browsers) to the Rails server to access resources like web pages, data, or services.
- Rails follows the **MVC (Model-View-Controller)** architecture, where requests are routed to specific controllers based on URL patterns.
- The **Router** (config/routes.rb) maps URLs to controller actions.
- Example: When a user visits /articles/1, Rails routes this GET request to the show action of the ArticlesController.

- Rails supports different HTTP methods:
  - GET (read data),
  - POST (create data),
  - PATCH/PUT (update data),
  - DELETE (delete data).
- Controllers receive the request, interact with models (data), and render views (HTML or JSON) as responses.

**c) Explain advantages of Ruby and Rails. [3]**

**Advantages of Ruby:**

1. Ruby has a clean, simple, and easy-to-read syntax that improves developer productivity.
2. It is highly flexible and supports multiple programming paradigms like procedural, object-oriented, and functional programming.
3. Ruby has a large standard library and a rich ecosystem of gems (libraries), speeding up development.

**Advantages of Rails:**

1. Rails follows the “convention over configuration” principle, minimizing setup and speeding up development.
2. It includes built-in features for database handling, testing, and security, which simplify web application development.
3. Rails promotes the DRY (Don’t Repeat Yourself) principle, helping to write maintainable and clean code.

---

➤ **NOV / DEC 2024**

**Q7)**

- a) **Explain the Ruby with respect to scalar types & their operations, control statements & arrays. [12]**  
→ already done , search ,see , study !
- b) **Write note on EJB. [5]**

**Enterprise JavaBeans (EJB):**

EJB is a server-side Java EE component architecture for building scalable, secure, and transactional enterprise applications. It manages system-level services like transactions, security, multithreading, and persistence, allowing developers to focus on business logic.

**Types of EJB:**

1. **Session Beans:** Handle business logic during client sessions.
  - *Stateless*: No client state maintained (e.g., currency conversion).
  - *Stateful*: Maintains client state (e.g., shopping cart).
  - *Singleton*: One shared instance for all clients (e.g., cache service).
2. **Entity Beans:** Represent persistent data (now replaced by JPA).
3. **Message-Driven Beans:** Handle asynchronous messaging via JMS queues/topics (e.g., order processing).

**Q8)**

a) Explain the Rails Applications & databases. Also explain Rails with AJAX. [12]

→ already done , search , see , study !!

b) Explain the advantages & Ruby & Rails. [5]

➔ Already done , search , see , study !!