1. **ruby on rails difference between render and redirect\_to..?**

**render**: calling a render creates a full response and sent back to the browser.

If you don’t include a call to render in your controller action then it is assumed you are trying to render the view of the same name as your controller action inside a folder of the same name as your controller.

If u want to render a different view ithin the same controller :

<%= render : action\_name%> and it will know where to look.

**If u want to rendr a different view from different controller:**

<%= render “viewname/controlleractionname”%>

like <%= render “home/navbar\_form”%>

inorder to debug th render, we can use redirect

**redirect\_to :** its tells the browser it need to makes a new request to different location, this could be a location within your application by calling something like

**redirect\_to widgets\_url,**

or it can be to a completely different website,

<%= redirect\_to <http://......,com> %> by default it uses the 302 rdirect , which is a temporary redirect. If you know any that traffic going to a certain place should always end up at a different place you should consider setting the status code to 301, which is a perminant redirect.

<%= redirect\_to <http://......,com> status:301 %>

**Best to use render i**nstead of redirect, beacause you shound not make browser need to make fresh call unless u want, a render doesnot make change the url of the page when u visit.

1. **Rake command:**

custom rake tasks have .rake extension file places in rails, we can create these custom tasks with the rails generate task command.

Task task\_name: [prerequisite\_task, : another\_task depend on] do

To pass arguments to your custom rake

task: task: taskname, [:arg\_1] => [:prerquisite\_1, :prerequisite\_2] do

[task, args], we can group tasks by placing them in namespaces.

http://guides.rubyonrails.org/command\_line.html

1. **Constructors:** it has 3 jobs:

It allocates space for object, it assigns instances variables their initial values, it returns instance of that class.

1. **Require and load method:**

When you want to use multiple files for code we use load and require methods

Require: loads the given file only once ex: require “filename”

**Load:** it loads named ruby source file every time it execute the method:

Require allows you to use number of extensions and libraries bundled from both the developers will **use load instead of require** because while in development mode when your using application on browser and making changes to the code at the same time, the changes are reloaded, overriding any caching behavior on the part of the web server.

REFER TO THE LINK: http://rubylearning.com/satishtalim/including\_other\_files\_in\_ruby.html

1. **MODULES AND MIXINS**:

Modules are much like classes in that they will hold collection of methods, constants abd other module and class definitions. Modules are defined same as classes just we write module keyword instead of class. Unlike class u cannot create object based on mudule nor can you subclass them, instead, you specify that u want functionalities of particular module to be added to the functionalities odf a class, or specific onject

2 benefits of modules:

1. Thhy acts as namespace that is every module name should be unique
2. They will allow you to share functionalities of different claasss.

**PARTIAL**: IT ALLOWS YOU TO ORGANISE AND REUSE THEview code in rails application, it starts with \_ and save as .html.erb. it can using render methos. there will global and localvariable partals.

**Default route:** We can set a default route by setting a root like root controllername/action in routes.rb in config

**Callback:** callback are methods gets called ata certain moment of objectlife cycle. With callback it is possible to write code and run when ever active record object is created, updated, deleted and validated from the database.

**CUCUMBER:**

Cucumber is a software tool used by computer programmers for testing other software. It runs automated acceptance tests written in a behavior driven development style. Cucumber is written in the ruby programming language. Cucumber projects are available for other platforms beyond ruby. Some use ruby cucumber with a bridge into the target language. **Capybara** which is part of cucumber testing framework, is an example of a web based test automation software.

**Rspec:**

Rspec is a behavior-driven development tool for ruby programming, bdd is an approach to software development that combines test-driven development, and domain-driven design, and acceptance test-driven planning. Rspec helps you to do tdd part of the equation and focus on the documentation and design aspects of of tdd

1. **how to implement BDD :-**
2. Start with a new Cucumber scenario. ...
3. Run the scenario and watch it fail. ...
4. Write a definition of the first failing or pending spec. ...
5. Test-drive the implementation of a Rails view using the red-green-refactor cycle with RSpec.

**polymorphic association**

is a term used in discussion of object relational mapping with respect to a problem of representing in a relational database domain and relationship from one class to multiple classes, in statistical languages like java all multiple classes are subclasses of superclass

**symbols : is a immutable which can be overwriten**

The "string" is a string, like as in any other language. The :**symbol** , is as you mentioned it's an efficient way of representing names and strings; they are literal values. It is initialized and exists only once during the **ruby** session. ... **Rails** methods uses hashes, thus, you find **symbols** a bit everywhere in **Rails**.

The truth of the matter is that **Symbols** are **Strings**, just with an important **difference**, **Symbols** are immutable. Mutable objects can be changed after assignment while immutable objects can only be overwritten. Ruby is quite unique in offering mutable**Strings**, which adds greatly to its expressiveness.

**cappuccino ruby on rails/:**

Cappuccino is completely server agnostic, meaning Cappuccino applications can be served using any HTTP server (for example Apache, lighttpd, Microsoft IIS, etc) and can communicate with any server side technology over HTTP (Ruby on Rails, Django, PHP, ASP, Java, CouchDB, etc). This lets you choose your server-side components based on whatever criteria is important to you (experience, existing infrastructure, etc).

**Reason for not to rely on rubygems:-**

<http://words.steveklabnik.com/how-to-not-rely-on-rubygemsorg-for-deployment>

its better to use bundler instead of using rubygems for deployment.

**gems for deployment :**

http://bundler.io/v1.13/deploying.html

<https://docs.travis-ci.com/user/deployment/rubygems/>

**gems for authentication:**

1. First, include the Devise gem in your Gemfile:

gem 'devise' 'version-if-u-want-any specific'

1. To install the newly-added gem, use:

bundle install

1. To install Devise, run:

rails g devise:install

and perform some settings manually, which are shown in the output of the command.

1. (Optional) For customization purposes, we can include the Devise gem's views in our app's views:

rails g devise:views

1. (Optional) Generate the user model, which will be used by Devise:

rails g devise user

1. Migrate your database:

rake db:migrate

1. You can see routes using:

rake routes

1. For signing up users, visit:

localhost:3000/users/sign\_up

**create database table in database with ruby on rails**

rails generate model mynewmodel name: string description: string

rake db:migrate

rails generate model content content\_id:auto-generated, law\_id:integer, parent\_id:integer, titel:string, text:string, content:string, [url:string](file:///C:\Users\mamid\Desktop\string)

**blocks and proc and lamda:**

**blocks:** blocks in which object rule is broken, blocks are language constructs and doesnot have any type. Block are handy small chunk of code which can be passed around easily to method to change the way of how they will behave. Block can be of two forms if u have small amount code use{} when u have large amount of data use do end.

Lamda: lamda are actually proc objects which slight differences. Lamda is a keyword and not a function which return an instance of class proc.

Lamda are kind of named blocks which can be passed around functionalities without conversions which is necessary for blocks.

Proc: proc is a short for procedure and it’s a class in ruby. We can define like

P = proc.new{ puts “hello” }

p.call

p.class

**difference btwn block, lamda and pro**: unlike block the other2 instance of proc class and have standardlone existence, block is a part of a syntax ofa method call. Which doesnot have standalone and will be appear in argument list. Block appears atmost in a argument list. In contrast u can pass multiple procs to methods.

Lamda and proc diss:

Lamdas check the number of arguments, while proc do not

**http://joycse06.github.io/blog/2014/12/exploring-rubys-proc-lambda-and-blocks/**

<http://joycse06.github.io/blog/2014/12/exploring-rubys-proc-lambda-and-blocks/>

**difference btwn blocks and proc and lamda:**

<http://awaxman11.github.io/blog/2013/08/05/what-is-the-difference-between-a-block/>

**filters:**

filters enables controllers to run pre and post processing code for its actions. These filters can be used to do authentication, caching and auditing the intended action is performed Or to do localization or output compression after the action has been performed. Filters have access to the request, response, and all the instance variables set by other filters in the chain or by the action (in the case of after filters).

Types of filters: mthod reference(symbol), external class, inline method(proc).

<http://api.rubyonrails.org/v2.3/classes/ActionController/Filters/ClassMethods.html>

git:

**Git** (/ɡɪt/) is a version control system (VCS) for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for software development, but it can be used to keep track of changes in any files.

**ruby gems**(**is a package managerfor ruby programming lang that provides a standard format for distributing ruby programs and libraries, is a tool designed to easily manage the installation of gems, and a srver for distributing them.)**

**GIT:**

Is an open sourse vrsion control, used for all kinds of web application development and several

Languages, it is created for rails later branched for other lang, it is created ffor speed and efficiency, it comes with rails installer.

**BENEFITS:** SPEED, DISTRIBUTED, DATA ASSURANCE, STAGING AND COLLABORATION.

**HEROKU:**

Is a cloud platform service that supports several programing lang. it’s a new way of building apps using git, it allows developers to focus on pure code as oppose to manage servers deployment etc.

Download the heroku toolkit which include git and bash tools, setup database, create git repository , push and ddeploy app in heroku

**html/databases questions**

1**)ruby names** refrs to classes, variables, methods and constants and modules used to create programs

This nameds distinguish other names used the specified rubynames and ruby refers to classes that are built and through which the objects are created.

2**)diff btwn symbol and string**:

The main difference in both will reflects in objctid, memoruy and processtime when both are in use

String is mutable so there will change only in assignments where synmbol will be overwritten.

String object will be written as “string objct jack”.object\_id#=>2250

3)**sessions: is** used to provide information about customes and servr side information

**Cookies** is used to store information with stored information on cliend and brwser

Session stores the user information and user actions, it will remains till the browser tillinformation is getting stored.

4) **request.xsl** is used to display controller which used to create ajax that is being handled by the new controller, it has Boolean values that r rturnde should generate only t/f

5)**mvc** provides flexibility and scalability used in many languages, controller is main part in this, which handles the request from other controller, and contacts views to provide response on behalf of request and also it contact to models, model will interact with database and provide rsponce to controller.

6)**components of mvc-**

**Validation:** it is essential component and used to validate to being put to intypes of streams like valida\_presence\_of, form-of

**Relationship**: it describes the relations brwn diff components and shows in form of has to one and has to many

**Callback** : it is used of respond when it is failure, it allows applicant to have some functionalities , before-save and aftersave.

**Validation group settings**: allows user to defin installed plugin settings

**Activerecord association relationship**: allows current record to active having relationship btwn 1 another.

7)ORM: is object relationship model, that models the classs and hlps in setting a relationship between existed model, it helps hlp in mapping the classes with tables in database , objects with rows and object attributes with columns. It shows relationship betwn objects and frmaes, it uses the model to display output.it keeps th data in db according to its relationships and perform the function accordingly.

On gems+ how to conncect database using gems

How do u deploy using gems

Crud operations

Scffholding

RUBY ON RAILS:

ITS AN OPEN SOURSE FRAMEWORK to develop an web application. By using this we can writ code vry easyand efficiently. W will devlop an application through MVC. To look and feelinside v use bootstrap, is an front end framework for developing responsive application.

The every question will goes through cycle like

User to router( routr is the url which the user inputs, it also have verbs like get which is used to translate into read operation an dother actions like subit the file delete something, router will figure out which controller should be incharge and passes the controls ,) to controller(this is the responsible for everything that the viw needs to rendor) to view(view is a template which which will send back to the userand like item of list, etc) to user.

There are some internals without knowing these internals we cannot develop application:

Install ruby:

Ruby –v(ruby 2.2)

Rails –v(rails 4.2)

Gem install rails

Rails new –h

Rails new nameof project(hu)

Cd hu/

Rails server (wb server will start Got to Chrome: type :localhost 3000)

Git init

Git add.

Rails s

New directory: generate controller

Rails generate controller home name of controller(index)

Goto chrome and type locolhost:3000/home/index

Sublime.

App-views-home-index.html.erb (go and as change welcome)