Ruby on Rails

CSCI-5448 : Object Oriented Analysis and Design

Submitted by :Dheeraj Chinni Ranga

What is Ruby on Rails

- Ruby on Rails is a web application framework written in Ruby, and it is a dynamic programming Language.
- Ruby on Rails uses MVC(Model view control) architecture pattern to organize application programming.

What is MVC ?

- Model maps to a table in database.
- View is a presentation of data in a particular format, triggered by a controller's decision to present the data. These are script based systems like PHP and are easy to access using Ajax.
- Controller responds to the external requests coming in to the application and sends responses to the external requests by determining which view to render.

Ruby

- Ruby is a pure Object Oriented Programming Language.
- Ruby is a Open-source server side scripting language similar to Perl and Python.
- Ruby is used to write Common Gateway Interface(CGI).
- Ruby is very scalable and big programs written in Ruby are easily maintainable.

Ruby

- Ruby supports multiple programming paradigms, including functional, object oriented, imperative and reflective.
- Ruby has a dynamic type system and automatic memory management.
- Ruby has rich in built functions that can be used directly in ruby scripts.
- Ruby is a Metaprogramming language.
- Ruby can be easily connected to DB2, MySQL, Oracle, and Sybase.

Sample Ruby code

- Ruby Class: Ruby shape class with height and width attributes.

```
Class Shape  # The name of the class should be capital.
    att_accessor :height, :width
    def initialize(height,widht) # Constructor of the class
        @height = height
        @width = width
    end
End
```

Sample Ruby code

- Creating an instance of Shape class:

```
Shape1 = Shape.new(20,10)
Shape2 = Shape.new(30,17)
```

Sample Ruby code:

- Method for Shape class:

  To calculate the area we add the area method in the shape class.

```
Def Area ()
    @height*@width
End
```

Sample Ruby Code

- Calling the method:

  To get the area we need to call the area method from the shape class.

```
S = Shape.new(10,5)
Area = S.Area()  # Calling area method
puts Area
Or
puts S.Area()
```

Rails

- Rails is a open source framework for developing database- backend web applications.
- Rails frame work has very rich functionalities which are extracted from the real world use cases.
- Everything in Rails is written in ruby except for configuring files- YAML.

Why Rails?

- Ruby is one of the best language for Meta programming and Rails uses this very well.
- The process of programming is much faster in Ruby because of the Object oriented nature and Library of open source code available with in the rails community.
- Rails projects will have the same structure and coding practices which helps the developers to move between the projects easily.

Why Rails?

- Rails has developed a strong focus on testing, and it has good testing frameworks.

- Rails and most of its libraries are open source, so there is no need to spend money buying these libraries

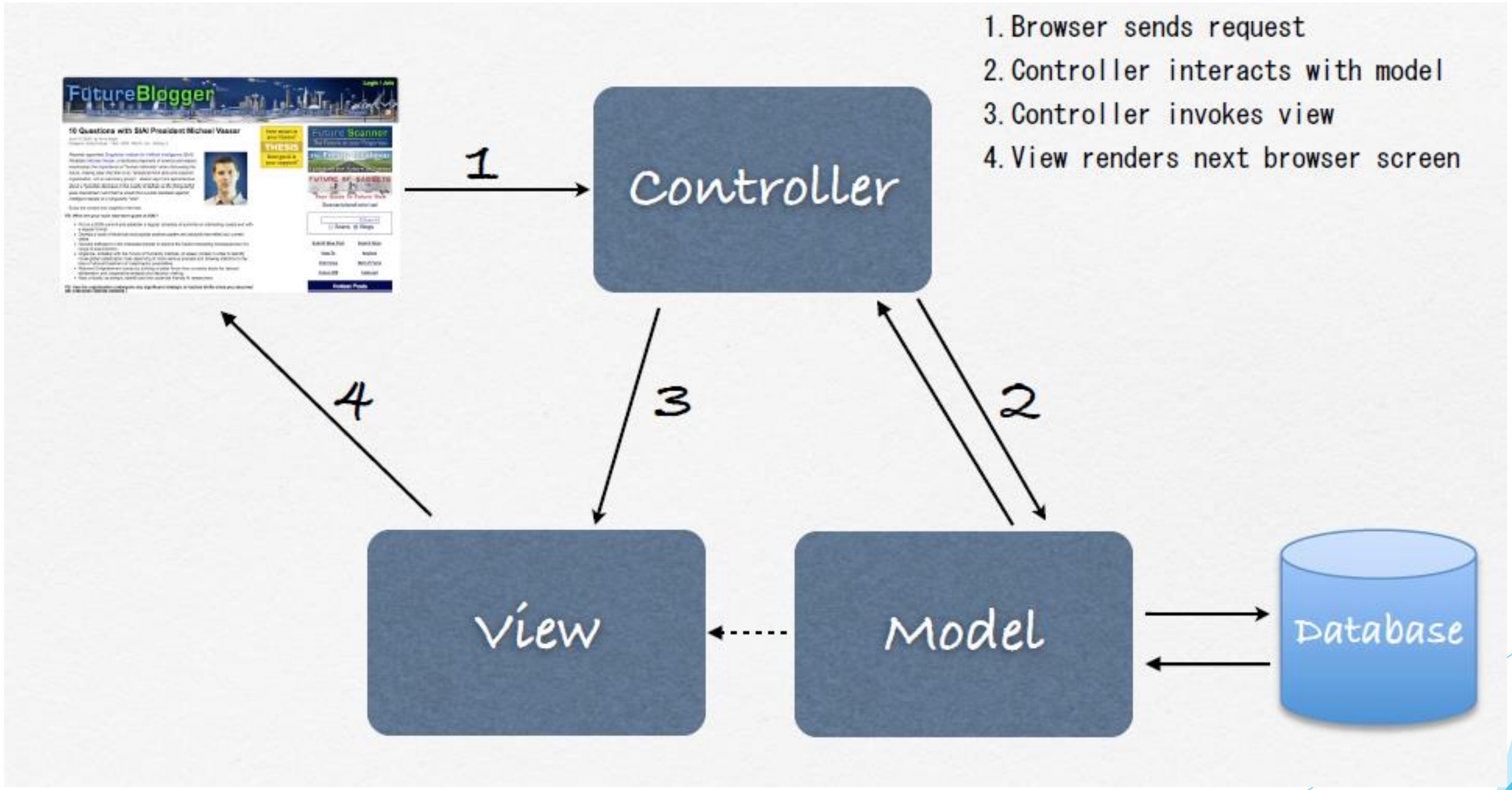
Why Rails?

- Active Record Framework:
  - Saves objects to the database.
  - Discovers the columns in database schema and automatically attaches them to domain objects using metaprogramming.
  - Naming convention allow database to discover specific field.

Why Rails?

- Active Pack:
  - This deals with the Action Controller and Action view.
  - Views gets the data from the controller.
  - Controller supplies data from the model according to the event given by the view.
  - Rails give a clear separation for control and presentation logic.

Rails MVC



1. Browser sends request
2. Controller interacts with model
3. Controller invokes view
4. View renders next browser screen

MVC

- Model:
  - This contains the data of the application
    - Transient
    - Stored(eg Database)
    - Enforces business rules of the application
    - Attributes
    - workflow

View

- Proves the user Interface
- Dynamic content rendered through templates

- Three Major types
  - Ruby in erb(embedded Ruby) templates
  - Xml.builder templates
  - rjs templates

Controller

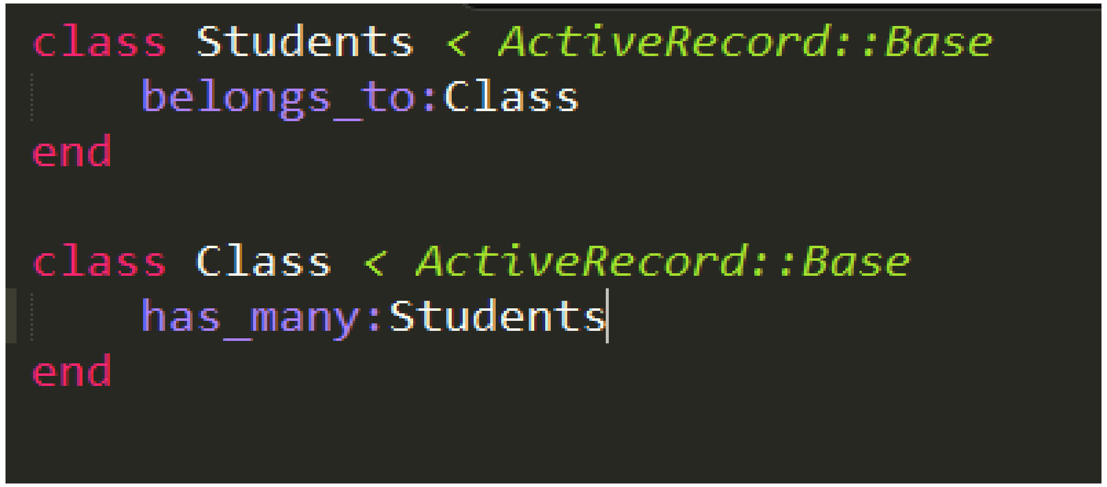
- Perform the bulk of the heavy lifting.
- Handles web requests
- Maintains session state
- Perform caching
- Manages helper Modules.

Example: Model

- Example to add student details:
- Initially create a Database with the name student_details using the sql query
  - **Create database student_details**
  - After doing this provide all the privileges on the Student_details database.
  - Create Active records that stores the instances of the database.
  - **Ruby script/generate model Name**
  - **Ruby Script/generate model Class**
  - These two commands generates two models in **Name.rb** and **Class.rb** in app/model folder.

Example

- Code snippet of the created models with active records association between the models.



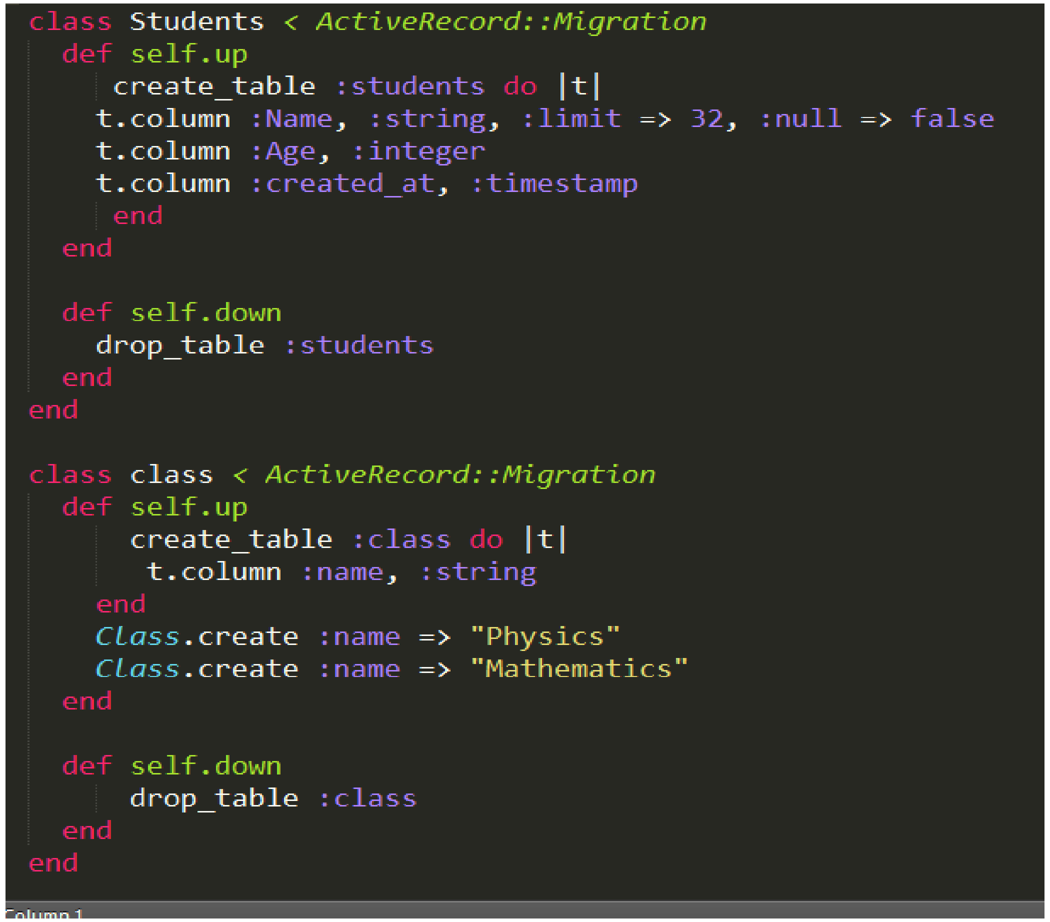
```
class Students < ActiveRecord::Base
    belongs_to:Class
end

class Class < ActiveRecord::Base
    has_many:Students
end
```

Example: Migration

- Create Migrations : Migration contains basic ruby syntax that describe data structure of database.
- Ruby script/generate Migration students
- Ruby script/generate Migration class
- After writing the column names in these files rub them using the command

  rake db:migrate

Example: Migration

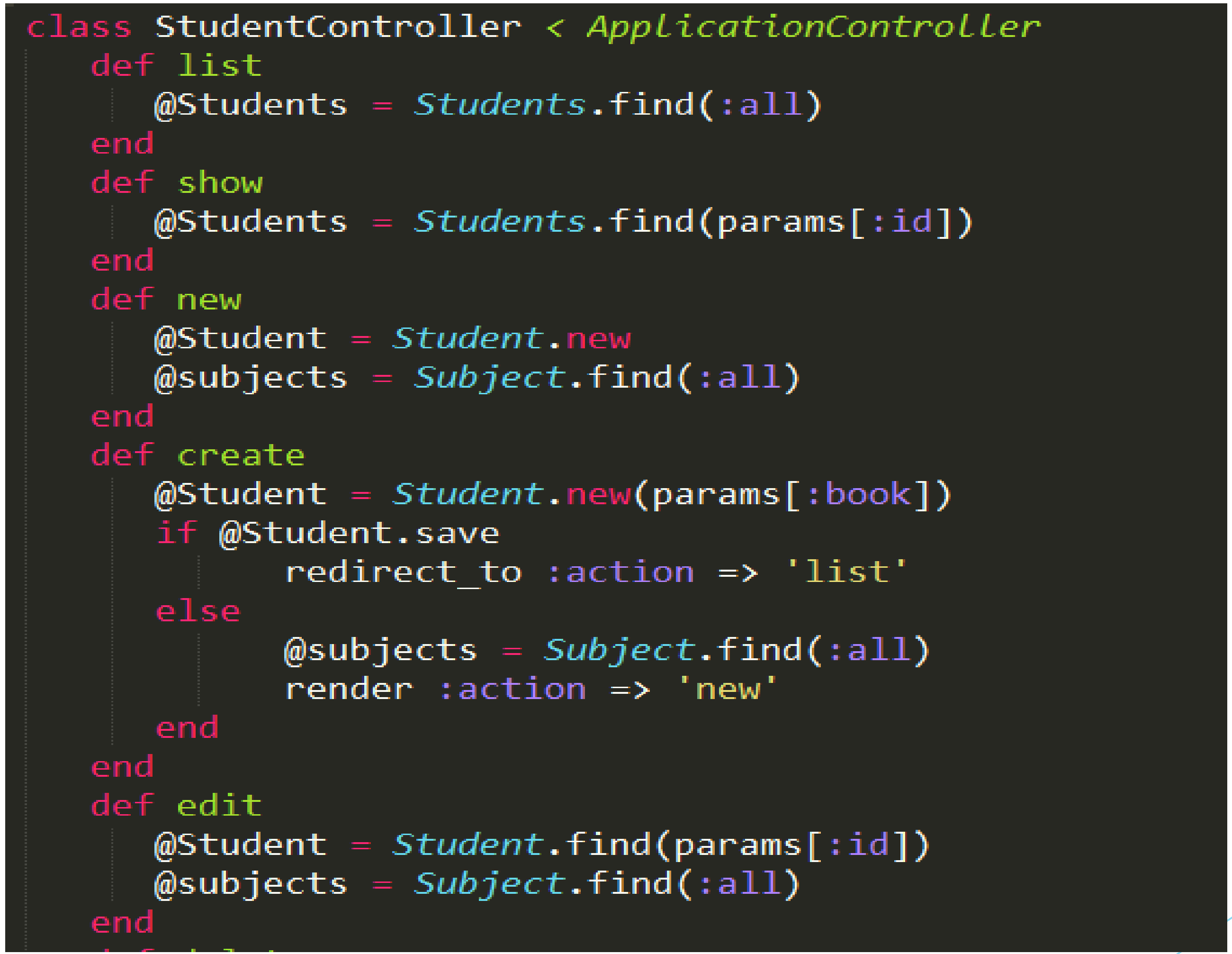


```ruby
class Students < ActiveRecord::Migration
  def self.up
     create_table :students do |t|
    t.column :Name, :string, :limit => 32, :null => false
    t.column :Age, :integer
    t.column :created_at, :timestamp
     end
  end

  def self.down
    drop_table :students
  end
end

class class < ActiveRecord::Migration
  def self.up
      create_table :class do |t|
       t.column :name, :string
    end
    Class.create :name => "Physics"
    Class.create :name => "Mathematics"
  end

  def self.down
      drop_table :class
  end
end
```

Example: Controller

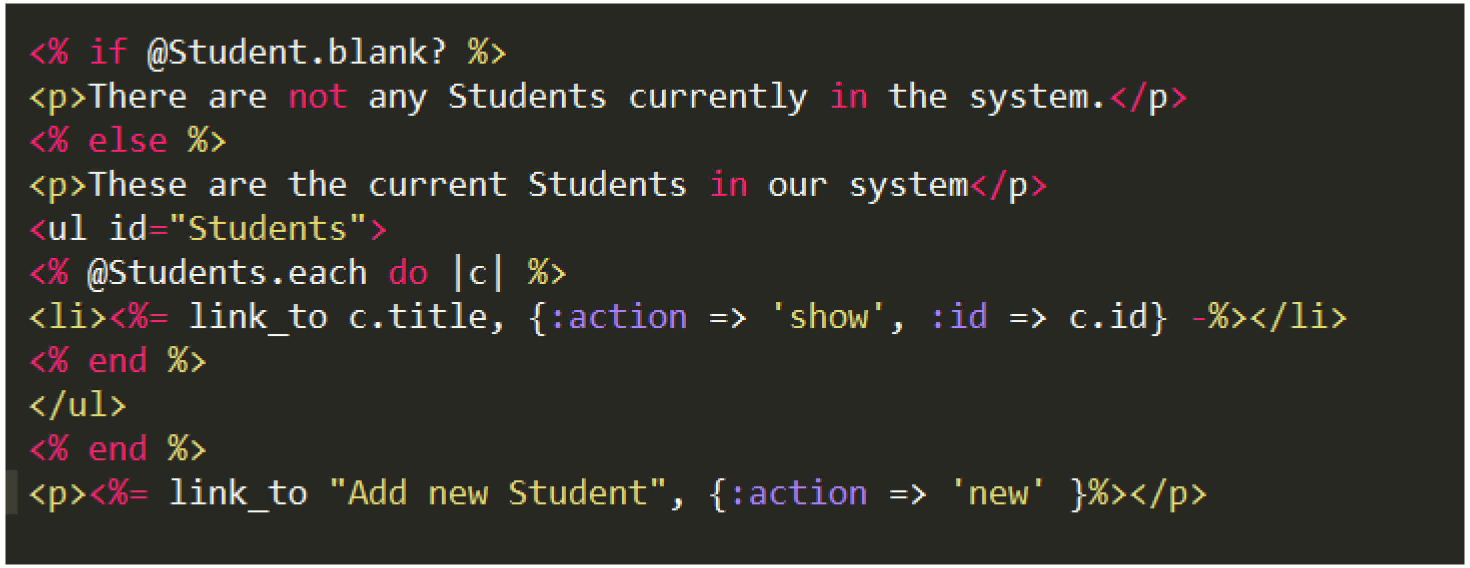
- Create a controller which communicates with both view and the model for the events in the view.
- Ruby script/generate controller student
- This will create a ruby file student in which we need to create methods for creating new, updating, showing and deleting the student records.

Example: Controller


```ruby
class StudentController < ApplicationController
   def list
      @Students = Students.find(:all)
   end
   def show
      @Students = Students.find(params[:id])
   end
   def new
      @Student = Student.new
      @subjects = Subject.find(:all)
   end
   def create
      @Student = Student.new(params[:book])
      if @Student.save
            redirect_to :action => 'list'
      else
            @subjects = Subject.find(:all)
            render :action => 'new'
      end
   end
   def edit
      @Student = Student.find(params[:id])
      @subjects = Subject.find(:all)
   end
```

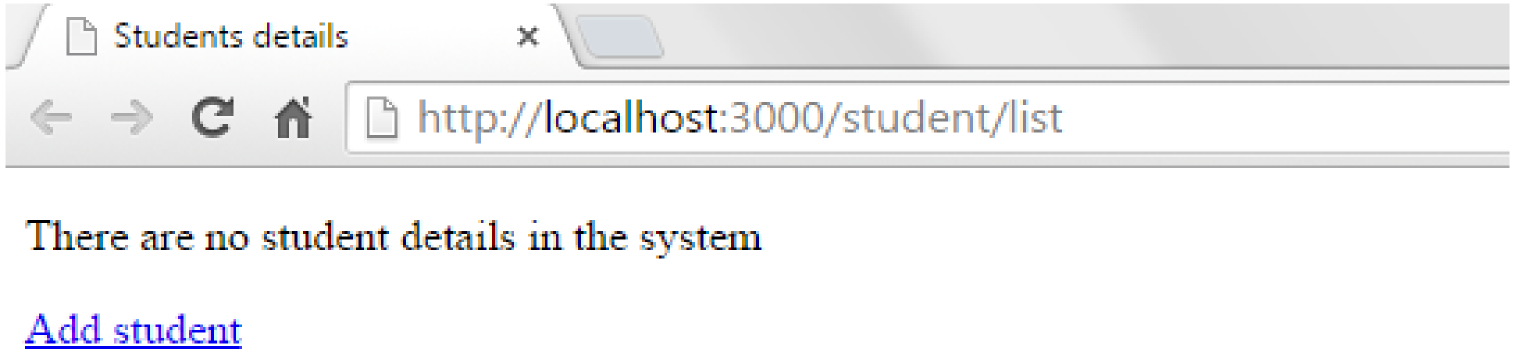
Example: View

- The last step is creating a view which is a HTML page with ruby script in it.
- The below code will show all the student details available in the system and it also provides a link to add new student.



```
<% if @Student.blank? %>
<p>There are not any Students currently in the system.</p>
<% else %>
<p>These are the current Students in our system</p>
<ul id="Students">
<% @Students.each do |c| %>
<li><%= link_to c.title, {:action => 'show', :id => c.id} -%></li>
<% end %>
</ul>
<% end %>
<p><%= link_to "Add new Student", {:action => 'new' }%></p>
```

Example: view

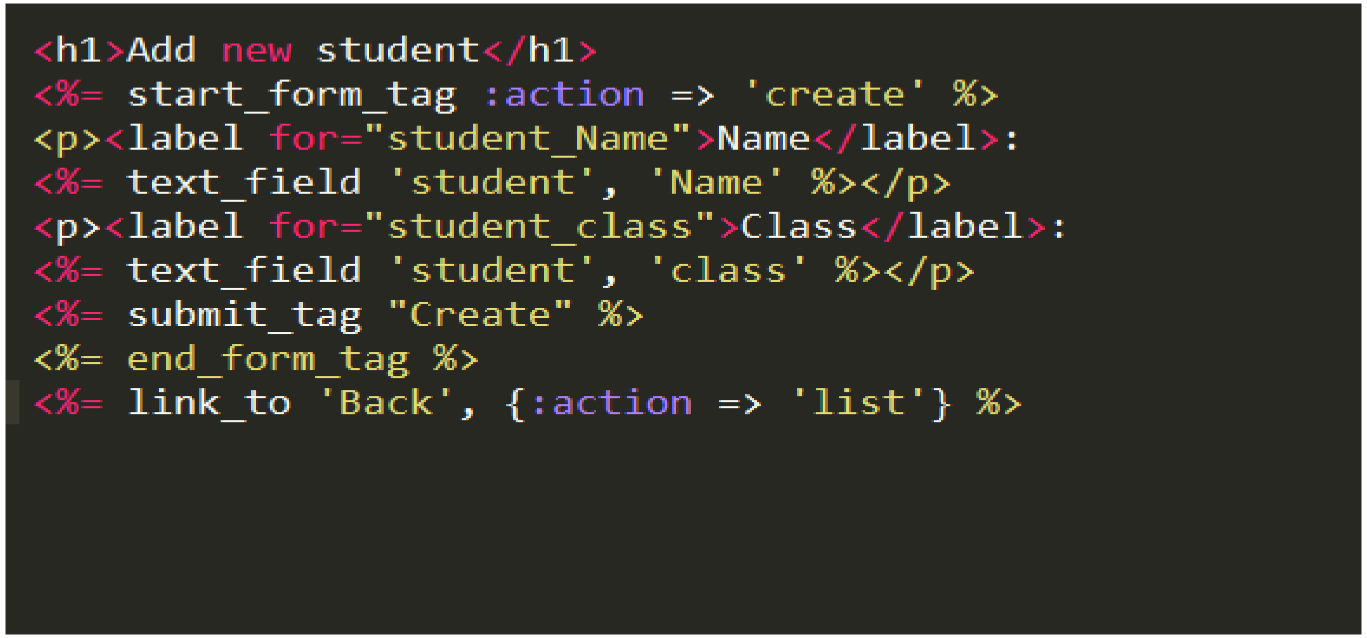


There are no student details in the system

Add student

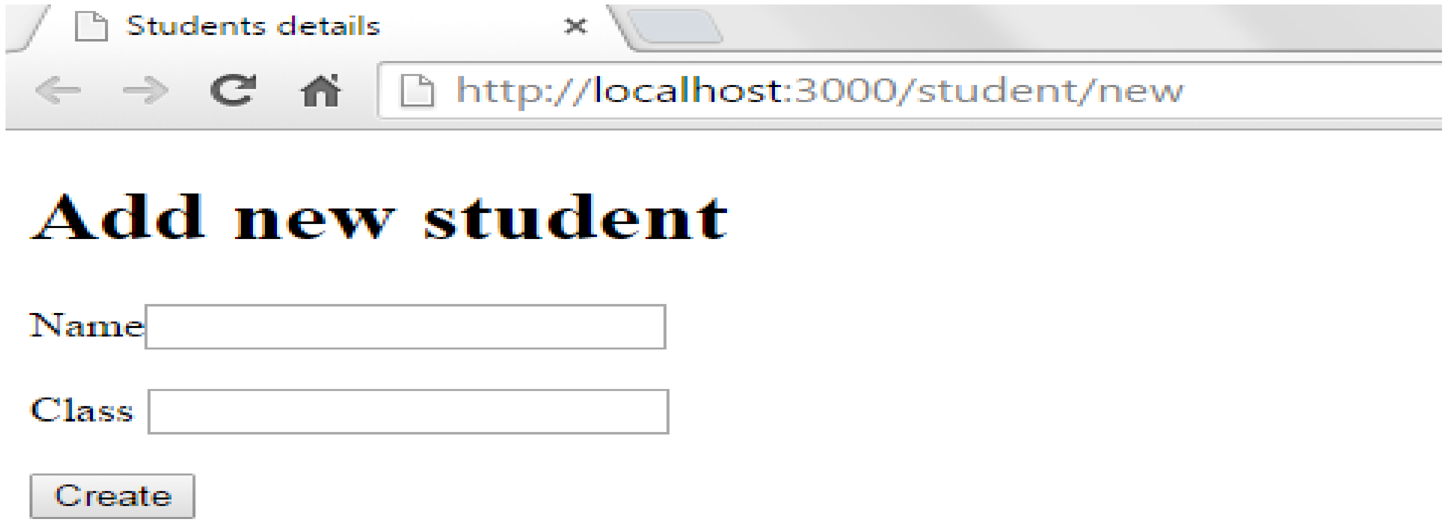
Example: view

- Adding student details in to the database.



```
<h1>Add new student</h1>
<%= start_form_tag :action => 'create' %>
<p><label for="student_Name">Name</label>:
<%= text_field 'student', 'Name' %></p>
<p><label for="student_class">Class</label>:
<%= text_field 'student', 'class' %></p>
<%= submit_tag "Create" %>
<%= end_form_tag %>
<%= link_to 'Back', {:action => 'list'} %>
```

Example: view



Students details ×

http://localhost:3000/student/new

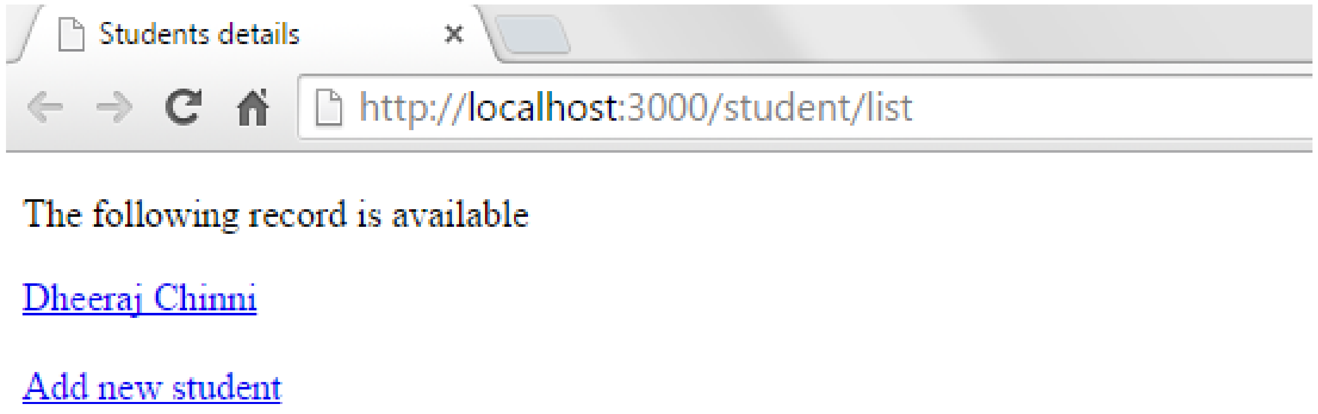
Add new student

Name

Class

Example: view

- The list after adding one record.



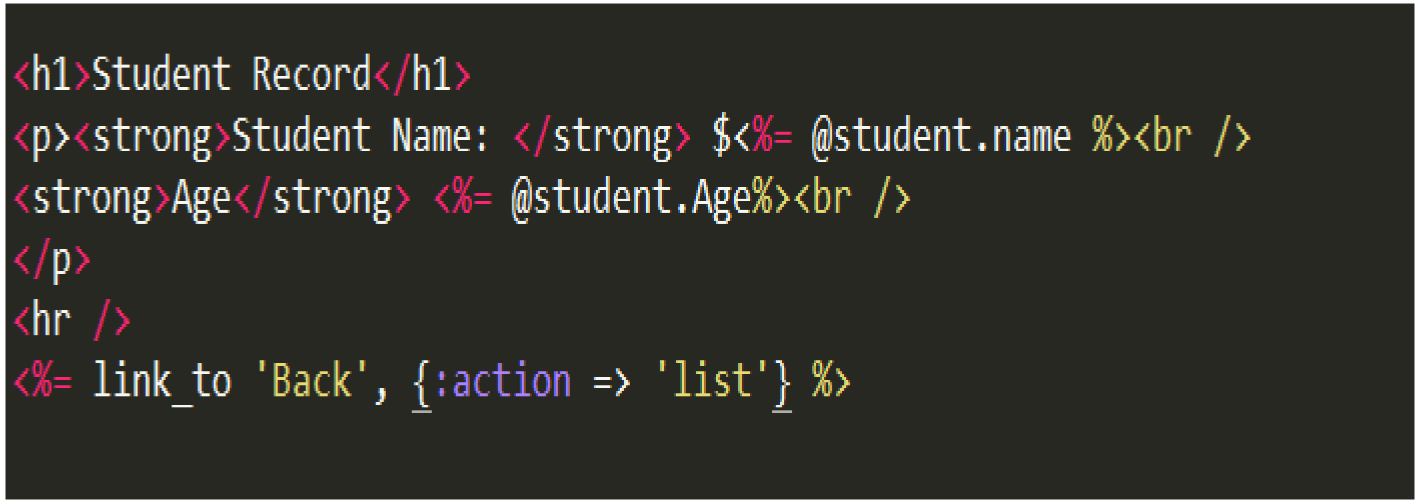
The following record is available

Dheeraj Chinni

Add new student

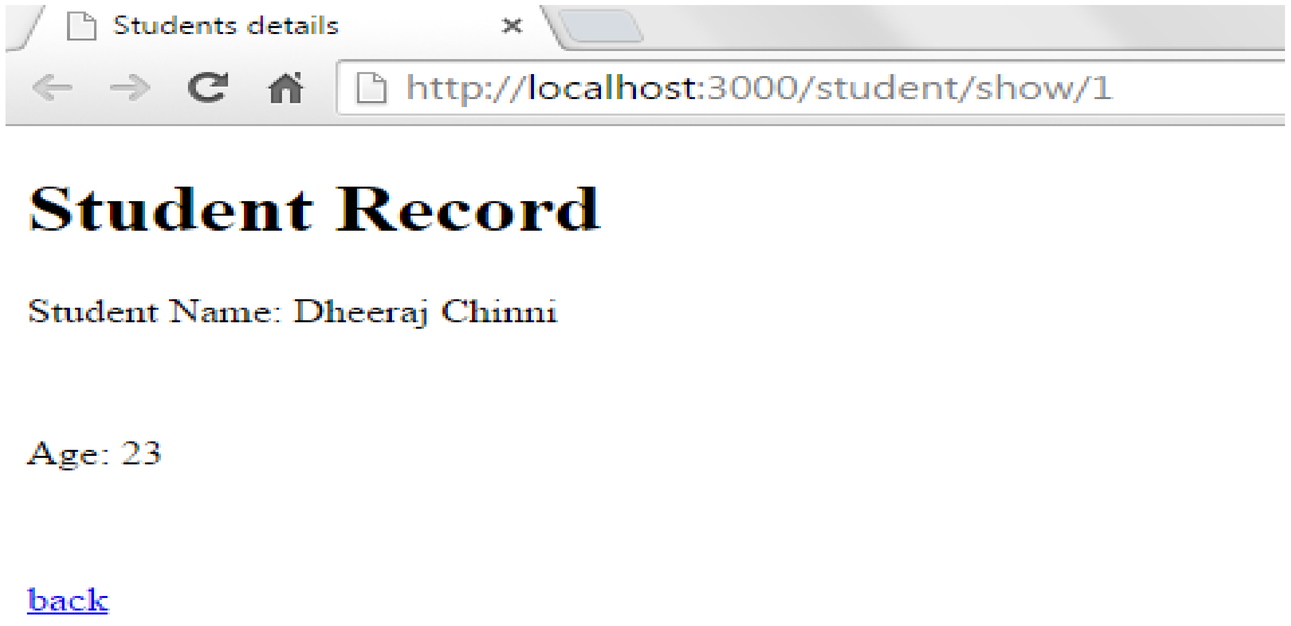
Example: View

- To display the details of the student.



```
<h1>Student Record</h1>
<p><strong>Student Name: </strong> $<%= @student.name %><br />
<strong>Age</strong> <%= @student.Age%><br />
</p>
<hr />
<%= link_to 'Back', {:action => 'list'} %>
```

Example: view


Students details

http://localhost:3000/student/show/1

Student Record

Student Name: Dheeraj Chinni

Age: 23

back

Disadvantages

- Runs Slowly compared to other Languages
- Installing and deploying is very confusing
- Very less expert Ruby Programming.
- Debugging is very slow
- No Clustering and two phase commit
- Compound Primary keys are not supported

Advantages

- It has built in testing, Migration and Some version control
- Very powerful, high level commands
- Easy to build prototypes and deploy them.
- MVC structure simple and easy to manage the files.
- Very less constrains compared to other frameworks.

References:

- http://rubyonrails.org/
- http://guides.rubyonrails.org/getting_started.html
- http://www.lynda.com/Ruby-Rails-tutorials.html
- http://betterexplained.com/articles/starting-ruby-on-rails-what-i-wish-i-knew/

Thank you ☺