CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | |

Internal Assessment Test 2– June. 2023

Sub:	Advanced Web Technologies							Sub Code:	20MCA41
Date:	24/6/2023	Duration:	90 min’s	Max Marks:	50	Sem:	IV	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
			CO	RBT
	PART I			
1	Write a note on Ruby methods	[10]	CO3	L2
	OR			
2	Explain input and output statements in ruby with suitable examples.	[10]	CO3	L2
	PART II			
3	Explain built-up methods for arrays and lists in ruby	[10]	CO3	L3
	OR			
4	Discuss all the string methods in Ruby	[10]	CO3	L3
	PART III			
5	What is Bootstrap? Discuss the bootstrap file structure	[10]	CO5	L2
	OR			
6	Discuss on bootstrap default grid system	[10]	CO5	L3
	PART IV			
7	Give an example how dynamic documents are generated in Ruby on rails	[10]	CO3	L4
	OR			
8	Explain with an example how static document are generated in Ruby in rails	[10]	CO3	L4
	PARTV			
9	Write Ruby Program to implement stack-like structure in an array	[10]	CO3	L5
	OR			
10	Write a Rails application program that accepts two integer values and produce the sum of two values and return it to the client	[10]	CO3	L6

Q1) Write a note on Ruby methods

- A method definition includes the method's header and a sequence of statements, ending with the end reserved word, which describes its actions.
- A method header is the reserved word def, the method's name, and optionally a parenthesized list of formal parameters.
- Method name must begin with lowercase letters.
- If the method has no parameters, the parentheses are omitted it means no parentheses.

Syntax:
```
def method_name
        statements….
     end
```

- You can represent a method that accepts parameters like this:

Syntax:
```
def method_name (var1, var2)
        Statements.....
      end
```

- You can set default values for the parameters which will be used if method is called without passing required parameters:

Syntax:
```
def method_name (var1=value1, var2=value2)
        statements….
     end
```

Example:

```
def test(a1="Ruby", a2="Perl")
     puts " programming language is #{a1}"
     puts "The programming language is #{a2}"
end
```

Local variables

- Local variables are either formal parameters or are variables created in a method.
- The scope of the local variable is from the header of the method to the end of the method.
- Local variables must be begin with either a lowercase letter or an underscore (_).
- Beyond the first character, local variable names can have any number of letters, digits, or underscores.

- The lifetime of the local variable is from the time it is created until end of th execution of the method.
- The return statement in ruby is used to return one or more values from a Ruby Method.

Example:

```
#!/usr/bin/ruby
def test
   i = 10          }
   j = 20          }
   k = 30          }  local variables
   z=#{i+j+k};     }
   return z
end

var = test          // function call
puts var
```

Parameters:

- In Ruby, parameters transmission of scalars is strictly one-way into the method.
- The value of the scalar actual parameters are available to the method through its formal parameters.

Example:

```
def swap(x,y)
   t = x
   x = y
   y = t
   puts "the values are #{x},#{y}"
end

a=1
b=2
swap(a,b)
```

Q2) Explain input and output statements in ruby with suitable examples.

Output is directed to the screen with the **puts** method (or operator).

Example:

```
>>name="Rohit"
=> "Rohit"
>>puts "my name is #{name}"
    my name is Rohit
=> nil
```

The value returned by puts is nil, and that value is returned after the string has been displayed.

Keyboard Input:

The **gets** method gets a line of input from the keyboard, default it will be taken as string.

Example:

```
>>fruit=gets mango
 => "mango\n"
```

If a number is to be input from the keyboard, the string from gets must be converted to an integer with to_i method.

If the number is a floating-point value, the conversion method is to_f.

Example:

```
>> age=gets.to_i
23
=> 23
>> age=gets.to_f
23.5
=> 23.5
```

Example Program:

Write a program to input four numbers a, b, c, x and find the value of the expression ax2+bx+c

```
#!/usr/bin/ruby
puts " enter the values of a"
     a=gets.to_i
 puts "enter the value of b"
     b=gets.to_i
puts "enter the value of c"
      c=gets.to_i
puts "enter the value of x"
      x=gets.to_i
result= a*x**2+b*x+c
puts " the value of the expression is: #{result}"
```

Q3) Explain built-up methods for arrays and lists in ruby

- **shift** method removes and returns the first element of the array.

Example:
```
>> ruby = [2,3,4,5,6]
=> [2, 3, 4, 5, 6]
>> first = ruby.shift
=> 2
>> ruby
=> [3, 4, 5, 6]
```

- **unshift** method takes a scalar or an array literal as a parameter. The scalar or arr literal is append to the beginning of the array.

Example:
```
>> ruby
=> [3, 4, 5, 6]
>> ruby.unshift ("mca","be")
=> ["mca", "be", 3, 4, 5, 6]
```

- **pop** method removes and returns last element from the array.
- **push** method also takes scalar or an array literal. The scalar or an array literal is added to the high end of the array.

Example:
```
>> ruby
=> ["mca", "be", 3, 4, 5, 6]
>> ruby.push(100,60)
=> ["mca", "be", 3, 4, 5, 6, 100, 60]
```

concat: If an array is to be catenated to the end of another array, concat is used.
Example:
```
>> rns1 = [1,2,3]
=> [1, 2, 3]
>> rns2 = [6,8,7]
```

reverse: The reverse method does what its name implies.
Example:
```
>> rns2
=> [6, 8, 7]
>> rns2.reverse
=> [7, 8, 6]
```

include?: The include? predicate method searches an array for a specific object.
Example:
```
>> rns2
=> [6, 8, 7]
>> rns2.include?(6)
=> true
```

sort: The sort method sorts the elements of an array.
Example:
```
>> rns3 = [90,54,23,12]
=> [90, 54, 23, 12]
>> rns3.sort
=> [12, 23, 54, 90]
```

Set operations
There are three methods that form perform set operations on two arrays.
& for set intersection
- for set difference
| for set union.

Example:

`>> set1 = [2,4,6,8]`	`>> set1 = [2,4,6,8]`	`>> set1 = [2,4,6,8]`
`=> [2,4,6,8]`	`=> [2,4,6,8]`	`=> [2,4,6,8]`
`>> set2 = [4,6,8,10]`	`>> set2 = [4,6,8,10]`	`>> set2 = [4,6,8,10]`
`=> [4,6,8,10]`	`=> [4,6,8,10]`	`=> [4,6,8,10]`
`>> set1 & set2`	**`>> set1 - set2`**	**`>> set1 | set2`**
`=> [4,6,8]`	`=> [2]`	`=> [2,4,6,8,10]`

Q4) Discuss all the string methods in Ruby

Method	**Action**	**Examples**
capitalize	Convert the first letter to uppercase and the rest of the letters to lowercase	`>>str="MCA"` `=> "MCA"` `>>str.capitalize` `=> "Mca"`
chop	Removes the last character	`>>str="MCA "` `=> "MCA "` `>>str.chop` `=> "MCA"`
chomp	Removes a newline from the right end, if there is one	`>>strg="mca\n"` `=> "mca\n"` `>>strg.chomp` `=> "mca"`
upcase	Converts all lowercase to uppercase	`>>str="Good"` `=> "Good"` `>>str.upcase` `=> "GOOD"`
downcase	Converts all uppercase to lowercase	`>>str="MCA"` `=> "MCA"` `>>str.downcase` `=> "mca"`
strip	Removes the spaces on both ends	`>>rnsit=" hello "` `=> " hello "` `>>rnsit.strip` `=> "hello"`
lstrip	Removes the spaces on the left end	`>>rnsit=" hello "` `=> " hello "` `>>rnsit.lstrip` `=> "hello "`
rstrip	Removes the spaces on the right end	`>>rnsit=" hello "` `=> " hello "` `>>rnsit.rstrip` `=> " hello"`
reverse	Reverse the characters of the string	`>>rnsit=" hello "` `=> " hello "` `>>rnsit.reverse` `=> " olleh "`
swapcase	Convert all uppercase letters to lowercase and all lowercase letters to uppercase	`>>rns="RaVi"` `=> "RaVi"` `>>rns.swapcase` `=> "rAvI"`

- **Catenation method**

The String method for catenation is specified by plus (+), which can be used as binaryoperator.

Example:
```
>> "Happy" + "" + "Morning"
 => "Happy Morning"
```

- **Assignment method**

Example:
```
>> a = "Happy"
 => "Happy"
>> b = a
 => "Happy"
>> b
=> "Happy"
```

- **<< method**

To append a string to the right end of another string, use the << method.

Example:
```
>> a = "Happy"
 => "Happy"
 >> a << "Morning"
 => "Happy Morning"
```

- **Chr method**

Ruby strings can be indexed, the indices begin at zero. The brackets of this method specify a getter method. The catch with this getter method is that it returns the ASCII code rather than the character. To get the character, the chr method must be used.

Example:

`>> be="civil"`	`>> be="civil"`
`=> "civil"`	`=> "civil"`
`>> be[2]`	`>> be[2].chr`
`=> 118`	`=> "v"`

- **Substring method**

A multicharacter substring of a string can be accessed by including two numbers in the brackets.

Example:

`>> city="bangalore"`	`>> name="Donald"`
`=> "bangalore"`	`=> "Donald"`
`>> city[3,3]`	`>> name[3,3]="nie"`
`=> "gal"`	`=> "nie"`
	`>> name`
	`=> "Donnie"`

- **== method**

The usual way to compare strings for equality is to use the == method as an operator.

Example:
```
>>"rnsit"=="rnsit"
=> true
>>"rnsit"=="bms"
=> false
```

- **< = > compare method**

It returns **-1** if the second operand is greater than the first

0, if they are equal

1, if the first operand is greater than second

Greater in this case means it belongs later in alphabetical order.

Example:
```
>>"apple"<=>"grape"
=> -1
>>"grape"<=>"grape"
=> 0
>>"grape"<=>"apple"
=> 1
```

- **repetition operator**

The repetition operator is specified with an asterisk (*). It takes a string as its left operand and an expression that evaluates to a number as its right operand.

Example:
```
>>"hello" * 5
=> "hellohellohellohellohello"
```

- **equal? method**

It determines whether its parameter is the same object as the one to which it is sent.

Example:
```
>> "snowstorm". equal? ("snowstorm")

=> false
```

This produces false because, although the contents of two string literals are the same, they are different objects.

- **eql? method**

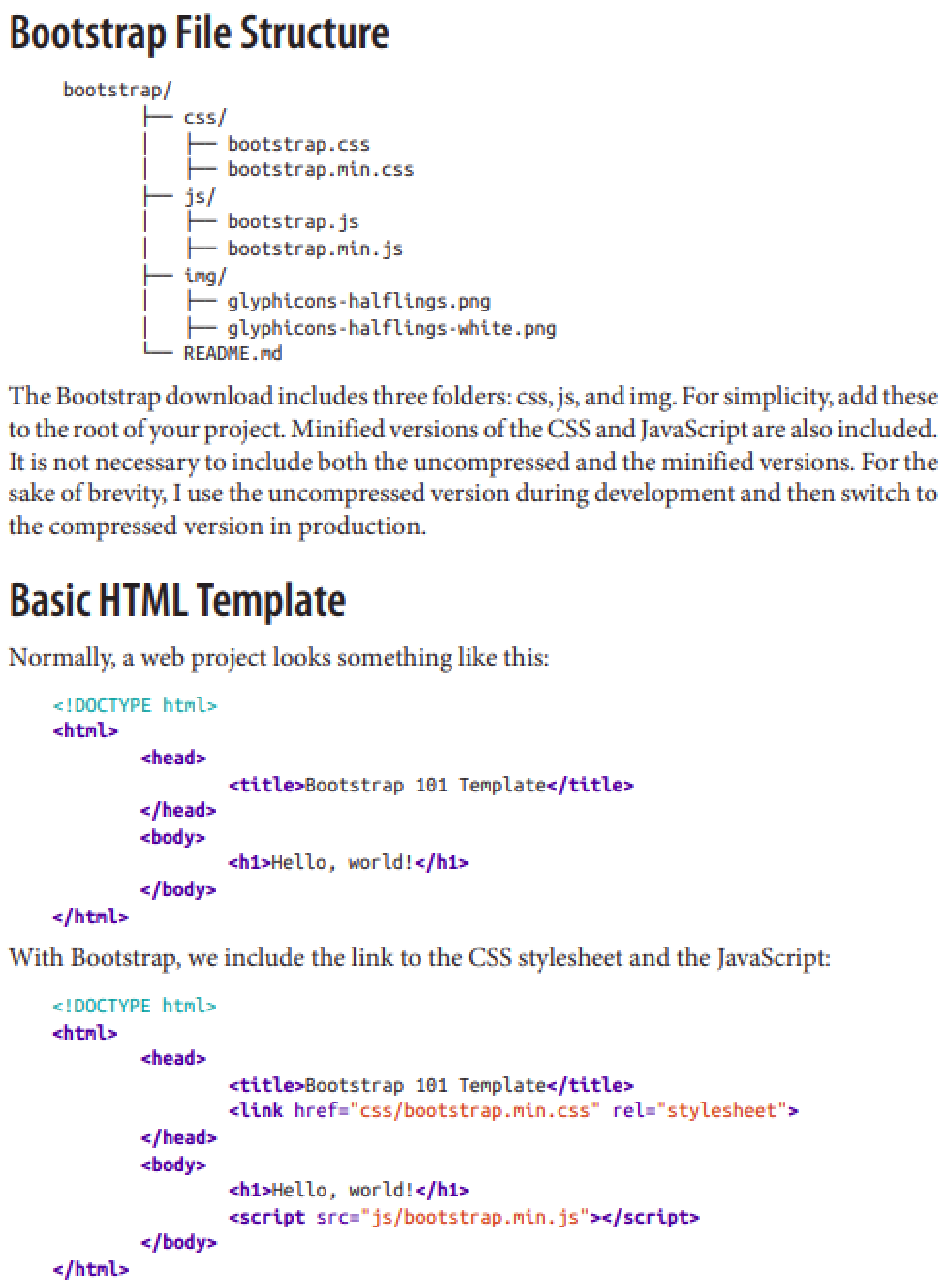
It returns true if its receiver object and its parameter have the same types and the same value.

```
Example: >> 7 ==7.0

         => true

         >> 7.eql?(7.0)

         => false
```

Q5) What is Bootstrap? Discuss the bootstrap file structure

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation and other interface components.

Bootstrap is an open source product from Mark Otto and Jacob Thornton who, when it was initially released, were both employees at Twitter. There was a need to standardize the frontend toolsets of engineers across the company



Bootstrap File Structure

```
bootstrap/
        ├── css/
        │   ├── bootstrap.css
        │   ├── bootstrap.min.css
        ├── js/
        │   ├── bootstrap.js
        │   ├── bootstrap.min.js
        ├── img/
        │   ├── glyphicons-halflings.png
        │   ├── glyphicons-halflings-white.png
        └── README.md
```

The Bootstrap download includes three folders: css, js, and img. For simplicity, add these to the root of your project. Minified versions of the CSS and JavaScript are also included. It is not necessary to include both the uncompressed and the minified versions. For the sake of brevity, I use the uncompressed version during development and then switch to the compressed version in production.

Basic HTML Template

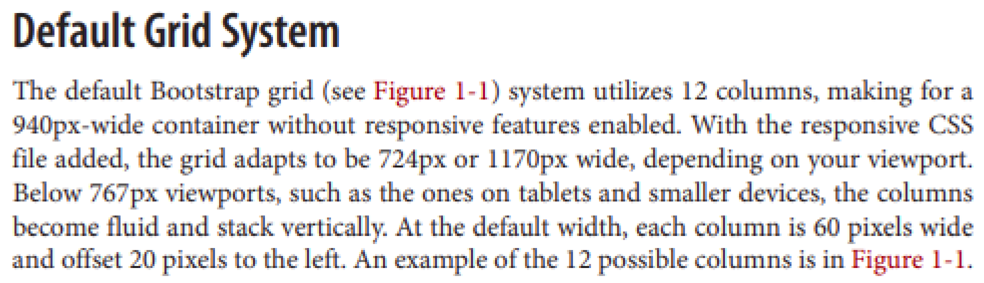
Normally, a web project looks something like this:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Bootstrap 101 Template</title>
        </head>
        <body>
                <h1>Hello, world!</h1>
        </body>
</html>
```

With Bootstrap, we include the link to the CSS stylesheet and the JavaScript:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Bootstrap 101 Template</title>
                <link href="css/bootstrap.min.css" rel="stylesheet">
        </head>
        <body>
                <h1>Hello, world!</h1>
                <script src="js/bootstrap.min.js"></script>
        </body>
</html>
```

Q6) Discuss on bootstrap default grid system



Default Grid System

The default Bootstrap grid (see Figure 1-1) system utilizes 12 columns, making for a 940px-wide container without responsive features enabled. With the responsive CSS file added, the grid adapts to be 724px or 1170px wide, depending on your viewport. Below 767px viewports, such as the ones on tablets and smaller devices, the columns become fluid and stack vertically. At the default width, each column is 60 pixels wide and offset 20 pixels to the left. An example of the 12 possible columns is in Figure 1-1.

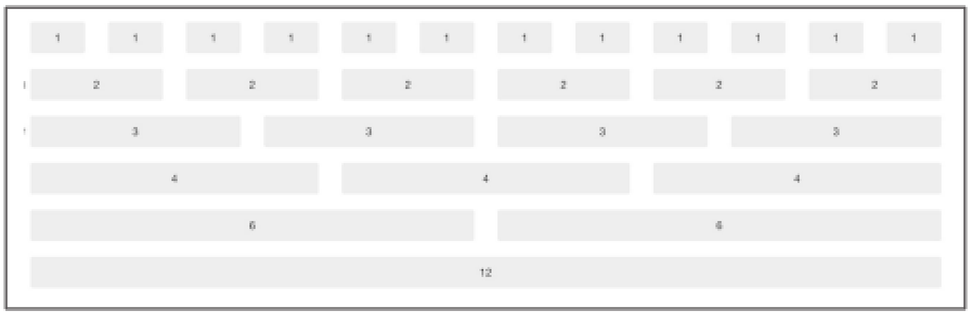


Figure 1-1. Default grid

Basic Grid HTML

To create a simple layout, create a container with a `<div>` that has a class of `.row` and add the appropriate amount of `.span*` columns. Since we have a 12-column grid, we just need the amount of `.span*` columns to equal 12. We could use a 3-6-3 layout, 4-8, 3-5-4, 2-8-2… we could go on and on, but I think you get the gist.

The following code shows `.span8` and `.span4`, which adds up to 12:

```
<div class="row">
  <div class="span8">...</div>
  <div class="span4">...</div>
</div>
```

Offsetting Columns

You can move columns to the right using the `.offset*` class. Each class moves the span over that width. So an `.offset2` would move a `.span7` over two columns (see Figure 1-2):

```
<div class="row">
  <div class="span2">...</div>
  <div class="span7 offset2">...</div>
</div>
```

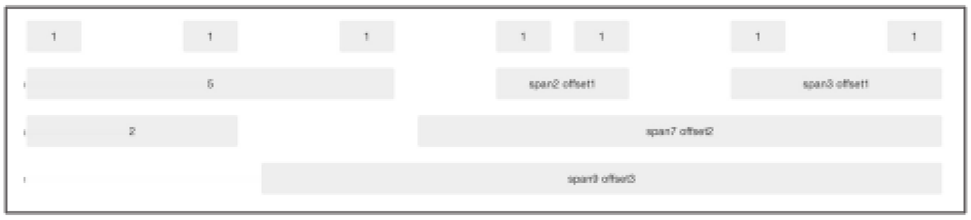


Figure 1-2. Offset grid

Nesting Columns

To nest your content with the default grid, inside of a `.span*`, simply add a new `.row` with enough `.span*` that it equals the number of spans of the parent container (see Figure 1-3):

```
<div class="row">
  <div class="span9">
    Level 1 of column
    <div class="row">
      <div class="span6">Level 2</div>
      <div class="span3">Level 2</div>
    </div>
  </div>
</div>
```

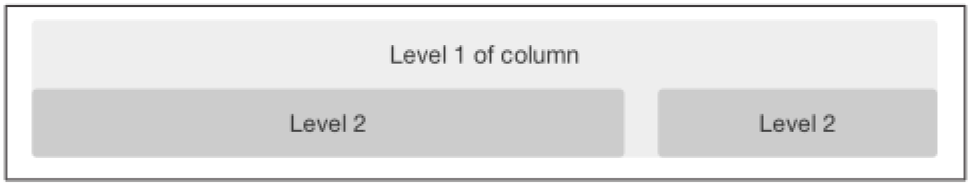


Figure 1-3. Nesting grid

Q7) Give an example how dynamic documents are generated in Ruby on rails

Dynamic Documents

- Rails offers three different approaches to producing dynamic documents. Here we discussed only one, which is to embed Ruby code in a template file. This is similar to some other approaches we have discussed, in particular PHP, ASP.NET and JSP.
- Ruby code is embedded in a template file by placing it between the <% and %> markers.
- If the Ruby code produces a result and the result is to be inserted into the template document, and equal sign (=) is attached to the opening marker.

  Example: The number of seconds in a day: <%=60 * 60 * 24%>
  Output: The number of seconds in a day: 86400

- The date can be obtained by calling Ruby's *Time.now* method, this method returns current day of the week, month, day of the month, time, time zone, and year, as a string.

  Example: <p>It is now<%= Time.now %></p>
  Output: It is now May 06 10:12:23 2015

[Note: The above (static document) procedure is same to develop a new rails application**s]**

1) Create a new directory: `> rails lab2`
2) Now change to new directory: `> cd lab2`
3) Generate a new Controller : `> ruby script/generate controller main`
4) Open timer Controller: `> notepad app\Controllers\main_controller.rb`

main_controller.rb

```
class MainController < ApplicationController
   def timer
         @t = Time.now
   end
end
```

5) Create a timer.rhtml under views\Main: `> notepad app\Views\Main\ timer.rhtml`

timer.rhtml

```
<DOCTYPE >
<html>
<head> <title> Simple Document </tittle>
</head>
<body>
<h1> Hello Welcome </h1>
<p> It is now <%= @t %> </p>
</body>
</html>
```

6) Start Ruby Server: `>ruby script/server`

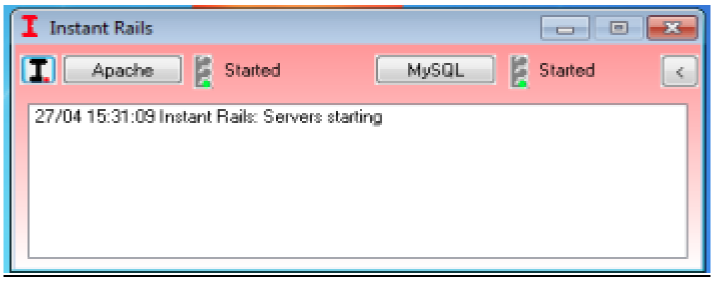
7) URL link is: `>http://localhost:3000/Main/timer`

Q8) Explain with an example how static document are generated in Ruby in rails

Static Documents

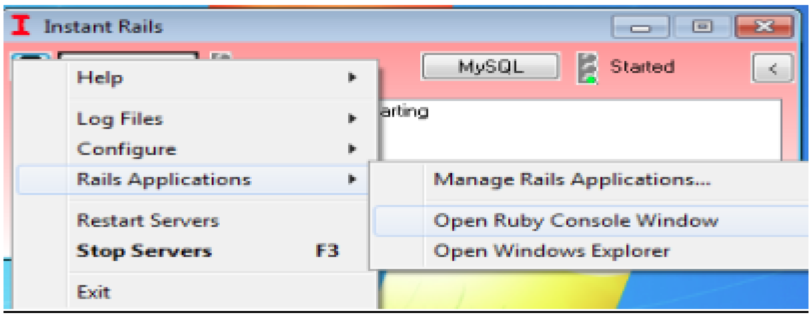
- This topic describes how to demonstrate the structure of the simplest possible Rails application and showing what files must be created and where they must be resided in the directory structure.

- First, you just click on Instant Rails icon. You will get the following small window.



27/04 15:31:09 Instant Rails: Servers starting

- Here, you click on black **I** on the left border of this window produces a small menu.



- Click the Rails Application entry in this menu, which opens another menu.
- Clicking on the Open Ruby Console Window entry in this menu opens a DOS command window in the following directory.

```
C:\InstantRails-2.0\rails_apps
```

- To this base directory, users usually add a new subdirectory for all their Rails applications.
- We named ours **lab1**. In the new lab1 directory, the new application rails is created with the following command.

```
rails lab1
```

- Rails responds by creating more than 40 files in more than 15 directories. This is part of the framework to support a Rails application. In this case **lab1**, 12 subdirectories are created.
- Now change to new directory, just typing: `cd lab1`
- The most interesting of which at this point is **app**. The **app directory** has four subdirectories-**models**, **views**, **controllers** and **helpers**. The **helpers** subdirectory contains Rails-provided methods that aid in constructing applications.
- One of the directories created by the rails command is **script**, which has several important Ruby scripts that perform services. This script creates two Ruby controller methods and also a subdirectory of the views directory where views code will be stored.
- **generate** is used to create part of an application controller.
- For our application, we pass two parameters to *generate*, the first of which is controller, which indicates that we want the *controller class* built. Second parameter is the

name we chose for the controller.

- The following command is given in the lab1 directory to create the controller.

```
ruby script/generate controller say
```

- Here **say** is the name of the **controller**. The response produced by the execution of this command as follows:

```
exists   app/controllers/ exists
app/helpers/ create app/views/say
create app/controllers/say_controller.rb
```

In the above output *exists* lines indicate files and directories that are verified to already exist.The *create* lines show the newly created directories and files.

- There are now two Ruby classes in the controllers directory, *application.rb* and *say_controller.rb.* where the say_controller.rb class is a subclass of application.rb.

Example:
```
class SayController < ApplicationController
end
```

- *Saycontroller* is an empty class, other than what it inherits from application.rb. The controller produces, at least indirectly, the response to requests, so a method must be added to SayController subclasses.

```
> notepad app\Controllers\say_controller.rb
```

say_controller.rb
```
class SayController < ApplicationController
  def hello
  end
end
```

- Next we need to build the view file which will be simple like XHTML file to produce the greeting. The view document is often called template. The following is the template for the lab1 applications.

```
> notepad app\Views\say\hello.rhtml
```

hello.rhtml
```
<DOCTYPE >
<html>
<head> <title> Simple Document </title>
   </head>
<body>
<h1> Hello Welcome </h1>
</body>
</html>
```

- The extension on this file is .rhtml, templates can include ruby code, which is interpreted by a ruby interpreter named ERb (Embedded Ruby), before the template is return to the requesting browser.

- Before the application can be tested, a rails web server must be started. A server is started with server script from the script directory.

```
ruby script/server
```

- Rails there are three different servers available. The default server is Mongrel, but apache and WEBrick are also available in rails.

- The default port is 3000. If a different port must be used, because 3000 is already being used by some other program on the system, the port number is given as a parameter to the server script.

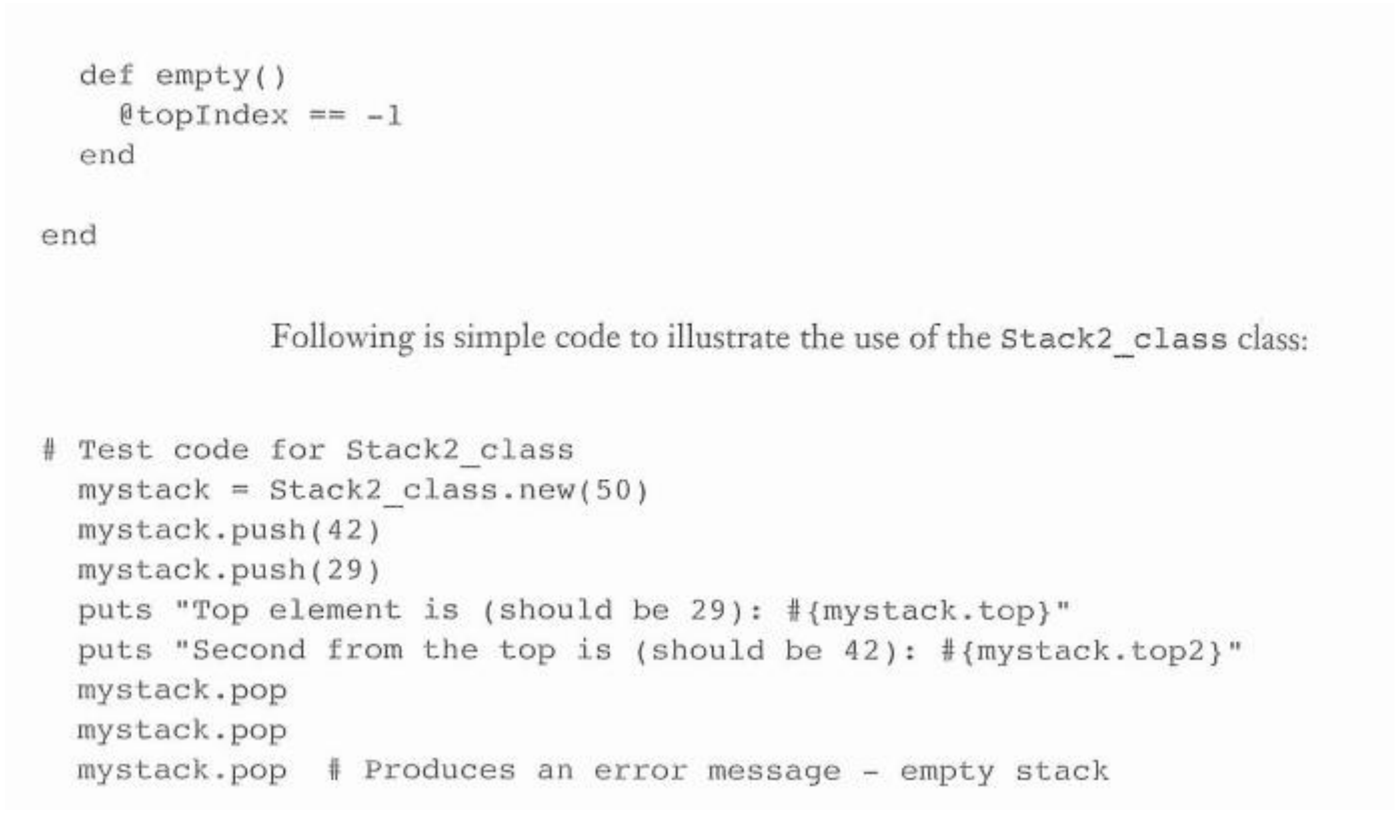
- The server name is always localhost because its running in the same machine on which applications resides.

- The complete URL of our application is: http://localhost/say/hello

Q9) Write Ruby Program to implement stack-like structure in an array

```
# Stack2_class.rb - a class to implement a stack-like
#                    structure in an array
class Stack2_class

# Constructor - parameter is the size of the stack - default is 100
```

```
  def initialize(len = 100)
    @stack_ref = Array.new(len)
    @max_len = len
    @top_index = -1
  end

 # push method
  def push(number)
    if @top_index == @max_len
      puts "Error in push - stack is full"
    else
      @top_index += 1
      @stack_ref[@top_index] = number
    end
  end

 # pop method
  def pop()
    if @top_index == -1
      puts "Error in pop - stack is empty"
    else
      @top_index -= 1
    end
  end

 # top method
  def top()
    if @top_index > -1
      return @stack_ref[@top_index]
    else
      puts "Error in top - no elements"
    end
  end

# top2 method
 def top2
   if @top_index > 0
     return @stack_ref[@top_index - 1]
   else
     puts "Error in top2 - there are not 2 elements"
   end
 end

# empty method
```



```
  def empty()
    @topIndex == -1
  end

end
```

Following is simple code to illustrate the use of the `Stack2_class` class:

```
# Test code for Stack2_class
  mystack = Stack2_class.new(50)
  mystack.push(42)
  mystack.push(29)
  puts "Top element is (should be 29): #{mystack.top}"
  puts "Second from the top is (should be 42): #{mystack.top2}"
  mystack.pop
  mystack.pop
  mystack.pop  # Produces an error message - empty stack
```

Q10) Write a Rails application program that accepts two integer values and produce the sum of two values and return it to the client

1. create new application using the following syntax
   ```
   $ rails new addition
   ```
2. Change the directory the following syntax
   ```
   $ cd addition
   ```
3. Creating a search controller which helps to search the books by using title as a search key.
   ```
   $ rails generate controller addnum result
   ```

Code

Addnum_controller.rb

```
Class Addnumcontroller<ApplicationController
        def result
                @num1=params[:num1].to_i
                @num2=params[:num2].to_i
                @res=@num1+@num2
        end
end
```

result.html.erb

```
<h1>Addition of two numbers</h1>
<form action=”result”>
<div>
```

```
<p>Enter Number:<input type=”text” name=”num1”></p><br/>
<p>Enter Number:<input type=”text” name=”num2”></p><br/>
<p><input type=”submit” value=”submit”></p><br/>
<p>Result:<%=@res%></p>
</div>
</form>
```