CS 317: Web Engineering
University of Balochistan
Handout 4: Lab 04: RoR first application

June 2016
Computer Science and IT
Instructor: Dr. Abdul Basit

First application

Creating application

Before generating a project in rails, we create a folder (a workspace) to store all web projects.

```
$ cd
$ mkdir railsapp
$ cd railsapp
```

Now, we can easily create a new project by a rails command.

```
$ rails new demo_app -d postgresql
$ cd demo_app
```

We will use the postgresql in this application. We can change to MySQL. If the -d switch is missing the default database is SQLite. “

Modeling Users

The users of our data model will contain information about users as there are many registration forms on the web.
Users of our app will have a unique integer identifier called **id** , a user **name** of type **string**, and an **email** address also a string.

The user resource

Now we will implement the user data model along with the web interface to that model. The user resource will allow us to think of users as objects that can be created, read, updated, and deleted. User resource will be created by a **scaffold** command.

```
$ rails generate scaffold User name:string email:string
```

If we are working with PostgreSQL, we have to create the database first by using rake command.

```
$ rake db:create
```

To proceed with application, we need to migrate the database using Rake. The rake simply updates the database with new user data model

```
$ rake db:migrate
```

```
Rake
Rake is Ruby make, a make-like language written in Ruby. Rails uses
Rake extensively, especially for the innumerable little
administrative tasks necessary when developing database-backed web
applications. The rake db:migrate command is probably the most
common, but there are many others; you can see a list of database
tasks using -T db:

$ rake -T db

To see all the Rake tasks available, run

$ rake -T

The list is likely to be overwhelming, but don't worry, you don't
have to know all (or even most) of these commands. By the end of Rails Tutorial, you'll know all the mos
```

Destroy or rollback the scaffold and database

If you done something wrong with the scaffold or database, we can destroy the scaffold and rollback the database
Rollback the database

```
$ rake db:rollback
```

Destroy scaffold

```
$ rails destroy scaffold User
```

Running the rails server

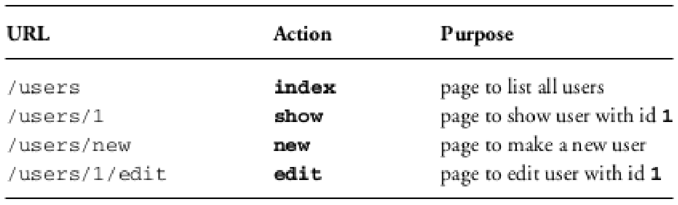
We can run the local web server using command

```
$ rails server
```

To check the output of your code, type the following code in the address bar of your favourite browser. You have to keep the rails server running while checking the pages on the browser.

```
$ http://localhost:3000
$ http://localhost:3000/users
```

Table 2.1 The correspondence between pages and URLs for the Users resource



URL	Action	Purpose
`/users`	**index**	page to list all users
`/users/1`	**show**	page to show user with id **1**
`/users/new`	**new**	page to make a new user
`/users/1/edit`	**edit**	page to edit user with id **1**

User tour

Routes

- rails routes config/routes.rb

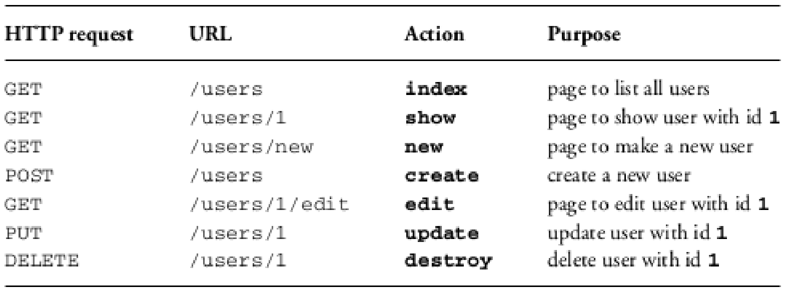
Controller Against every url there is a specific action (a function) in controller.

- app/controllers/users_controller.rb

View Now you should be able to understand this file

- app/views/users/index.html.erb

REST



HTTP request	URL	Action	Purpose
GET	/users	**index**	page to list all users
GET	/users/1	**show**	page to show user with id 1
GET	/users/new	**new**	page to make a new user
POST	/users	**create**	create a new user
GET	/users/1/edit	**edit**	page to edit user with id 1
PUT	/users/1	**update**	update user with id 1
DELETE	/users/1	**destroy**	delete user with id 1

Model View Controller (MVC)

Modeling posts

The posts model is simpler only needs **id** as integer and content as **string**. We need an additional field **user_id** for linking to the user table.

Generating posts

We can also use scaffold command for the post to generate posts resources.

```
$ rails generate scaffold Post content:text user:references
```

Now use Rake to migrate the Post table to the database.

```
$ rake db:migrate
```

The command above will update the database by creating the table Post.

Validates contents

To add a validation to the contents of the Post model. Open `app/models/post.rb` and write the following code

app/models/post.rb

```
class Post < ActiveRecord::Base
  validates :content, presence: true
end
```

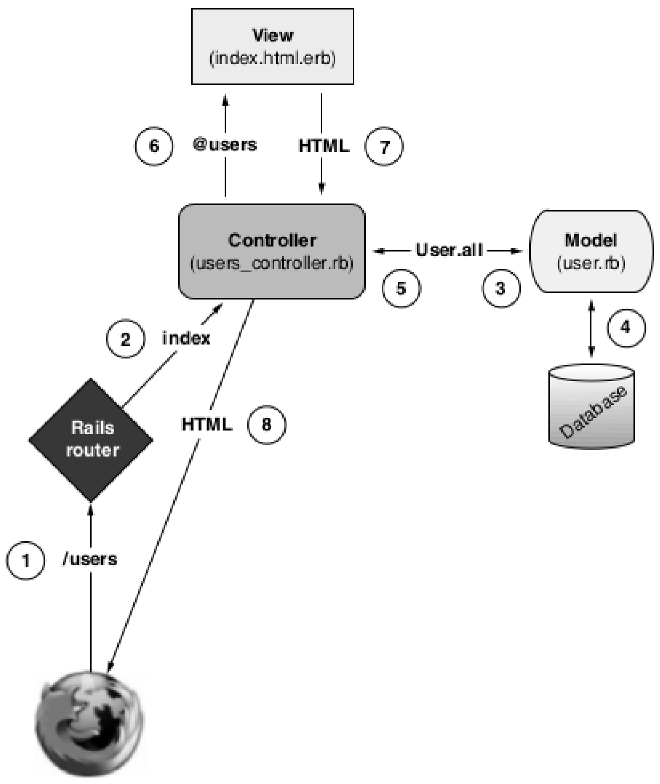
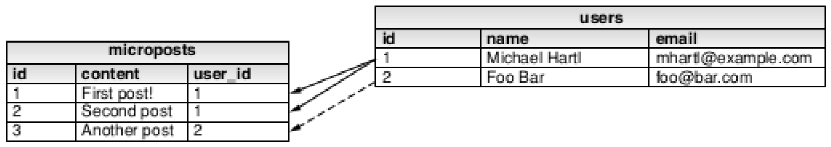


Figure 1: A detailed diagram of MVC in rails.



microposts		
id	content	user_id
1	First post!	1
2	Second post	1
3	Another post	2

users		
id	name	email
1	Michael Hartl	mhartl@example.com
2	Foo Bar	foo@bar.com

Figure 2: Association between users and posts.

Database entity associations

We can associate the tables in the database using rails models. Modify the user and post model to create association in database.

app/models/user.rb

```
class User < ActiveRecord::Base
  has_many :posts
end
```

app/models/post.rb

```
class Post < ActiveRecord::Base
  belongs_to :user

  validates :content, presence: true
end
```

Postgres Commands

Some sample commands to see your database in postgreSQL. To connect to the postgreSQL we do

```
$ sudo -u postgres psql
```

To see the list of databases

```
# \l
```

To connect to the database

```
# \c <database name>
# \c demoapp_development
```

To see the contents of a database

```
# SELECT * from a <table name>
# SELECT * from a Users;
```

To exit from the database

```
# \q
```

Rails console

Rails console is the command line tool to access your models and do changes in your database using the class objects.

```
> u = User.find(1)
> u.posts
> u2 = User.first
> u = User.all
> u[0]
> u[0].posts
```

Also perform some CRUD (create, read, update, delete) operations using rails console.

CRUD (Create, Read, Update, Delete)

```
Create
  > u = User.new
  > u.name = "Akmal Khan"
  > u.save
Read
  > User.find(3)
Update
  > u = User.find(2)
  > u.email = "hamid@gmail.com"
  > u.save
Delete
  > u = User.find(2)
  > u.destroy
```