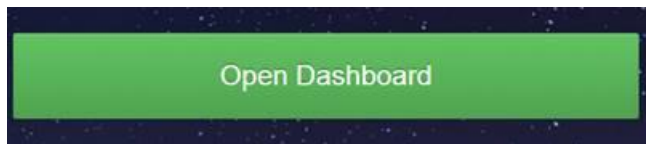


## Class Notes: CLT Part Time Class 08

We'll continue working in the Nitrous Dashboard, which gives us a complete toolchain for Ruby and Rails. You can walk through all the instructions below from Chrome, ***no need to open any other windows or programs on your computer***. Easy!

Open Chrome and navigate to Nitrous at <https://nitrous.io>

Make sure you are logged in with your Nitrous account, then click on the big green 'Open Dashboard' button:

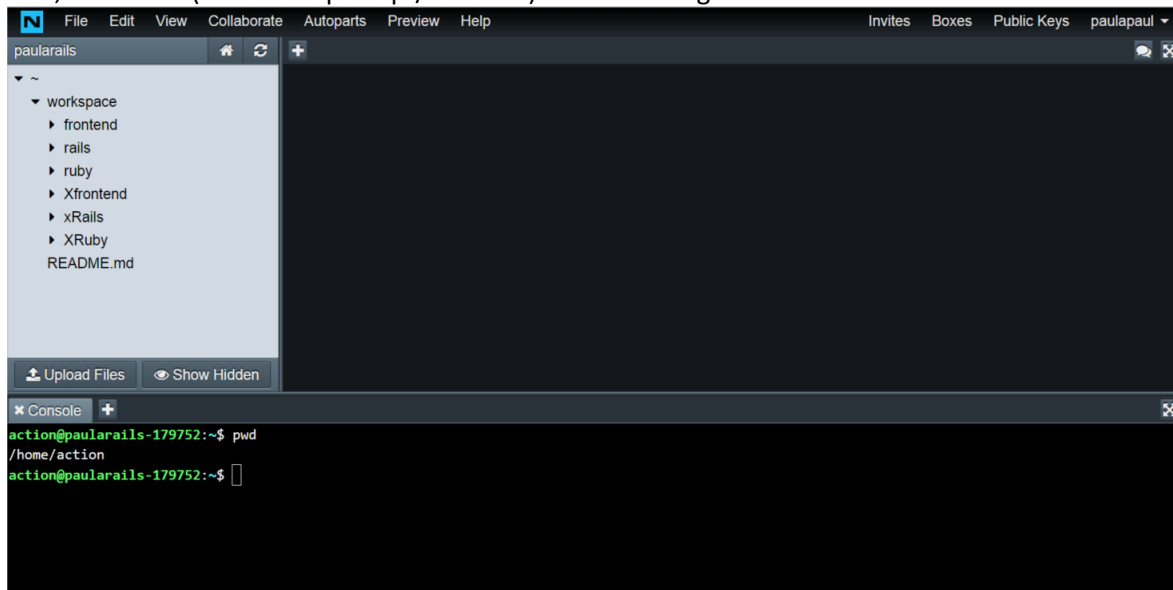


Wake up your Nitrous box and open the IDE (refer to our previous class notes if you need some tips on this part).

You should see your workspace and your three working folders in the top area of the browser window:

- frontend
- rails
- ruby

And, a Console (command prompt/terminal) window along the bottom:



Remember in the last class we used the **pwd** command to figure out the current working directory in the console? Use the following command to make your rails folder the working directory, and let's code.

**cd workspace/rails**

## Going to School with Forms, Parameters and Paging

Of course, we'll build a new app! This one is called 'RailsU' (short for 'Rails University'), and is modeled after the [Microsoft ASP.NET MVC 5 Contoso University sample](#) to illustrate how your Rails MVC skills can translate to other frameworks and languages.

1. First, make sure you are working in the Rails working directory (check your command prompt and use 'cd' to change to workspace/rails if needed).

```
action@paularails-179752:~/workspace/rails(master)$
```

2. Ok, let's go. Create the new RailsU app by entering:

**rails new RailsU**

3. Once you see the command prompt, what's next? Scaffold!

Before you scaffold, let's do some domain modelling. Here's what our RailsU app will manage:

- **Instructor**
  - last\_name:string
  - first\_name:string
  - hire\_date:date
  - *has\_many :departments*
- **Student**
  - last\_name:string
  - first\_name:string
  - enrollment\_date:date
  - *has\_many :enrollments*
- **Department**
  - name:string
  - budget:float
  - start\_date:date
  - instructor\_id:integer
  - *belongs\_to :instructor* (the Administrator)
  - *has\_many :courses*
- **Course**
  - number:integer
  - title:string
  - credits:integer
  - department\_id:integer
  - *belongs\_to :department*
  - *has\_many :enrollments*
- **Enrollment**
  - grade:string
  - student\_id:integer
  - course\_id:course
  - *belongs\_to :student*
  - *belongs\_to :course*

4. Now that your Rails app has been created, don't forget to change your working directory so that you are entering scaffold commands within the app directory:

**cd RailsU**

5. Ok – our domain model matches up pretty closely to the model from Contoso University sample (see the Note below for more details). Go ahead and scaffold by entering the following commands (press Enter after each):

```
rails g scaffold Instructor last_name:string first_name:string hire_date:date
rails g scaffold Student last_name:string first_name:string enrollment_date:date
rails g scaffold Department name:string budget:float start_date:date instructor_id:integer
rails g scaffold Course number:integer title:string credits:integer department_id:integer
rails g scaffold Enrollment grade:string student_id:integer course_id:integer
```

**Note** for those who like to go ‘into the weeds’... the Contoso University provides an example of model inheritance (Students and Instructors both inherit from a Person, similar to the ruby example we covered a few classes ago). This is also possible in Rails. If you would like to learn more about how to do this using Single Table Inheritance in Rails, check out this very nice write up of why you might want to do this, along with some tips and guidelines: <http://www.alexreisner.com/code/single-table-inheritance-in-rails>

6. This takes care of the resource scaffolding for our models (with associated views and controllers). Let’s scaffold a home and about page, to be routed through a single ‘Pages’ controller. To do this, enter

```
rails g controller Pages home about
```

7. Don’t forget to create the back end database for your scaffold. Enter:

```
rake db:migrate
```

8. Now that we have all the MVC files for RailsU, let’s add the associations to the model files. In Nitrous, open **app/models/course.rb** and add the associations:

```
1 class Course < ActiveRecord::Base
2   belongs_to :department
3   has_many :enrollments
4 end
```

9. Going down the line, open **app/models/department.rb** and add the associations:

```
1 class Department < ActiveRecord::Base
2   belongs_to :instructor
3   has_many :courses
4 end
```

10. Keep going... open **app/models/enrollment.rb** and add the associations:

```
1 class Enrollment < ActiveRecord::Base
2   belongs_to :student
3   belongs_to :course
4 end
```

11. Almost done... open `app/models/instructor.rb` and add the associations:

```
1 class Instructor < ActiveRecord::Base
2   has_many :departments
3 end
```

12. Last but not least, open `app/models/student.rb` and add the associations:

```
1 class Student < ActiveRecord::Base
2   has_many :enrollments
3 end
```

13. **Don't forget to save all your models!** Now, using our knowledge of `seeds.rb`, let's load some data using those models:

Open `app/db/seeds.rb` and enter:

```
1 # This file should contain all the record creation needed to seed the database with its default values.
2 # The data can then be loaded with the rake db:seed (or created alongside the db with db:setup).
3 #
4 Instructor.create!(last_name: 'Paul', first_name: 'Paula', hire_date: Date.new(1983,4,20))
5
6 Student.create!(last_name: 'Cool', first_name: 'Joe', enrollment_date: Date.new(2014,6,1))
7 Student.create!(last_name: 'Hacker', first_name: 'Jane', enrollment_date: Date.new(2014,6,1))
8
9 Department.create!(name: 'Code Slinging', budget: 750, start_date: Date.new(2014,1,1), instructor_id: Instructor.last.id)
10
11 Course.create!(number: 1111, title: 'The Joy of Code', credits: 3, department_id: Department.last.id)
12 Course.create!(number: 2222, title: 'Zen Debugging', credits: 3, department_id: Department.last.id)
13
14 Department.create!(name: 'Cat Herding', budget: 50, start_date: Date.new(2014,1,1), instructor_id: Instructor.last.id)
15
16 Course.create!(number: 3333, title: 'Message Spinning', credits: 3, department_id: Department.last.id)
17 Course.create!(number: 4444, title: 'Ego Wrangling', credits: 3, department_id: Department.last.id)
18
19 Enrollment.create!(grade: 'A', student_id: Student.last.id, course_id: Course.first.id)
20 Enrollment.create!(grade: 'A', student_id: Student.last.id, course_id: Course.last.id)
21 Enrollment.create!(grade: 'B', student_id: Student.first.id, course_id: Course.first.id)
22
```

Or, copy this from my github repo, at

<https://github.com/PaulaPaul/RailsU/blob/master/db/seeds.rb>

14. Once you have the information saved in your `seeds.rb` file, enter the following command to populate your database with the records:

**rake db:seed**

15. While we are modifying our app files, let's add a 'root route' that shows the courses we offer (we'll change this to the home page later).


Open `app/config/routes.rb` and insert the root route at the top of the file, above the resources

```
1 Rails.application.routes.draw do
2   root to: 'courses#index'
3
4   get 'pages/home'
5
6   get 'pages/about'
7
8   resources :enrollments
```

16. Everything on the 'back end' is all wired up! Now we just need to make it look pretty, by giving our controllers more information that they can then use within views, and adding a CSS framework like bootstrap. Preview what you have so far:

**rails s -b 0.0.0.0**

17. Preview 3000 to see the Course index (our 'root route'):



Number	Title	Credits	Department	
1111	The Joy of Code	3	3	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
2222	Zen Debugging	3	3	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
3333	Message Spinning	3	4	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
4444	Ego Wrangling	3	4	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>

[New Course](#)

18. For now, just add the following strings to the end of your browser address bar to see the additional information that was seeded in your RailsU app (we'll add some better navigation later)

.....nitrousbox.com/departments

.....nitrousbox.com/students

.....nitrousbox.com/instructors

.....nitrousbox.com/enrollments

19. Great! Now, to make this app look a little better, let's add bootstrap and a simple navigation bar. You may have some code for this in the application.html.erb file in your Music\_DB app, or feel free to copy it from my github repo:

<https://github.com/PaulaPaul/RailsU/blob/master/app/views/layouts/application.html.erb>

20. open `app/views/layouts/application.html.erb` and add the bootstrap headers and nav bar for your app:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Rails University</title>
5   <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.css" type="text/css">
6   <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap-theme.min.css" type="text/css">
7   <%= stylesheet_link_tag 'application', media: 'all', 'data-turbolinks-track' => true %>
8   <%= javascript_include_tag 'application', 'data-turbolinks-track' => true %>
9   <%= csrf_meta_tags %>
10  <script src="//code.jquery.com/jquery-2.1.3.js"></script>
11  <script src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.js"></script>
12 </head>
13 <body>
14   <nav class="navbar navbar-default navbar-fixed-top" role="navigation">
15     <!-- Brand and toggle get grouped for better mobile display -->
16     <div class="container-fluid">
17       <div class="navbar-header">
18         <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
19           <span class="sr-only">Toggle navigation</span>
20           <span class="icon-bar"></span>
21           <span class="icon-bar"></span>
22           <span class="icon-bar"></span>
23           <span class="icon-bar"></span>
24           <span class="icon-bar"></span>
25         </button>
26         <%= link_to Rails.application.class.parent_name, root_url, :class => "navbar-brand" %>
27       </div>
28
29       <!-- nav links, forms, and other content for toggling -->
30       <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
31         <ul class="nav navbar-nav">
32           <li><%= link_to "About", pages_about_url %></li>
33           <li><%= link_to "Students", students_url %></li>
34           <li><%= link_to "Courses", courses_url %></li>
35           <li><%= link_to "Instructors", instructors_url %></li>
36           <li><%= link_to "Departments", departments_url %></li>
37         </ul>
38       </div><!-- /.navbar-collapse -->
39     </div><!-- /.container-fluid -->
40   </nav>
41
42   <div class="container">
43     <%= yield %>
44   </div>
45
46 </body>
47 </html>
```

21. If you preview the app now, you'll see the cool, responsive, adaptive Bootstrap nav bar, but the container is a little too high and hidden under the nav bar. Let's fix that with some CSS.

One quick way to do this is to replace the default CSS in your scaffolds.css file.

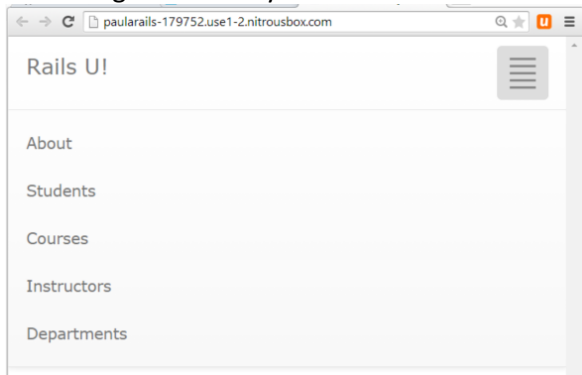
Open `app/assets/stylesheets/scaffolds.css` and replace it with the following code:

```
1 body {
2   padding-top: 50px;
3   background-color: #fff;
4   color: #333;
5   font-family: verdana, arial, helvetica, sans-serif;
6   font-size: 13px;
7   line-height: 18px;
8 }
9
10 /* universal */
11
12 html {
13   overflow-y: scroll;
14 }
15
16 section {
17   overflow: auto;
18 }
19
20 textarea {
21   resize: vertical;
22 }
23
24 .center {
25   text-align: center;
26 }
27
28 .center h1 {
29   margin-bottom: 10px;
30 }
```

22. Now, preview your app – looking much better!



23. Don't forget to resize your browser and make sure the hamburger is working (my favorite part):



24. Now, to get rid of those unsightly integer 'department\_id' numbers and make the course listing look a little more interesting.

Getting rid of the department\_id number is easy. We want to show the department.name instead, so thanks to our model we can change the course index to refer to

**course.department.name**, instead of **course.department\_id**

Open the **app/views/courses/index.html.erb** file and change one line:

**course.department\_id** to **course.department.name**

```
<tbody>
<% @courses.each do |course| %>
  <tr class="row">
    <td><%= course.number %></td>
    <td><%= course.title %></td>
    <td><%= course.credits %></td>
    <td><%= course.department.name %></td>
    <td><%= link_to 'Show', course,:class => "btn-primary btn-sm" %></td>
    <td><%= link_to 'Edit', edit_course_path(course),:class => "btn-primary btn-sm" %></td>
    <td><%= link_to 'Destroy', course,:class => "btn-warning btn-sm", method: :delete,
                                data: { confirm: 'Are you sure?' } %></td>
  </tr>
<% end %>
</tbody>
```

While you are there, do some styling.

Add some bootstrap buttons and include some bootstrap classes on the table. Insert some bootstrap panels for the heading, body and footer, and voila!

Rails U!AboutStudentsCoursesInstructorsDepartments

Courses

Number	Title	Credits	Department			
1111	The Joy of Code	3	Code Slinging	Show	Edit	Destroy
2222	Zen Debugging	3	Code Slinging	Show	Edit	Destroy
3333	Message Spinning	3	Cat Herding	Show	Edit	Destroy
4444	Ego Wrangling	3	Cat Herding	Show	Edit	Destroy

New Course

If you'd like to copy the styling from my github repo, the course index.html.erb file can be found here: <https://github.com/PaulaPaul/RailsU/blob/master/app/views/courses/index.html.erb>



25. We haven't touched our controllers yet, so let's do something to bring them into play. The Contoso University sample shows the courses (enrollments) associated with a student. Let's see what it takes to do that in Rails.

To get started, open the **app/views/students/show.html.erb** file and add some bootstrap styling, plus a section where we'll add a table of the student's enrollments:

```
1 <p id="notice"><%= notice %></p>
2 <div class="panel panel-default">
3   <div class="panel-heading">
4     <h3 class="panel-title">Student Details</h3>
5   </div>
6   <div class="panel-body">
7     <div>
8       <dl class="dl-horizontal">
9         <dt>Last name</dt>
10        <dd><%= @student.last_name %></dd>
11        <dt>First name</dt>
12        <dd><%= @student.first_name %></dd>
13        <dt>Enrollment Date</dt>
14        <dd><%= @student.enrollment_date %></dd>
15        <dt>Enrollments</dt>
16        <dd>
17          <table class="table">
18
19          </table>
20        </dd>
21      </dl>
22    </div>
23  </div>
24  <div class="panel-footer">
25    <%= link_to 'Edit', edit_student_path(@student),:class => "btn-primary btn-sm" %>
26    <%= link_to 'Back', students_path,:class => "btn-primary btn-sm" %>
27  </div>
28 </div>
```

To include the enrollments on the Show view for the student, we need to make the Student controller aware of the enrollments and courses.

26. Open the **app/controllers/students\_controller.rb** file and add one line under the 'show' method (remember, this is the method – or action – that controls the **show** view for a student) Add the following line to the **show** method:

**@enrollments = Enrollment.where(student\_id: @student.id)**

This will make sure we only show the enrollments associated with the selected student.

```
10 # GET /students/1
11 # GET /students/1.json
12 def show
13   @enrollments = Enrollment.where(student_id: @student.id)
14 end
```

27. Now we can include the enrollments associated with a student, on the student's show view. Add the code to fill in the table in your `app/views/students/show.html.erb` file.

```
14 <dd><%= @student.enrollment_date %></dd>
15 <dt>Enrollments</dt>
16 <dd>
17 <table class="table">
18 <thead>
19 <tr class="row">
20 <th>Course Title</th>
21 <th>Grade</th>
22 </tr>
23 </thead>
24 <tbody>
25 <% @enrollments.each do |enrollment| %>
26 <tr class="row">
27 <td><%= enrollment.course.title %></td>
28 <td><%= enrollment.grade %></td>
29 </tr>
30 <% end %>
31 </tbody>
32 </table>
33 </dd>
```

Check out the way we can get the course title via the enrollment (using `enrollment.course.title`). This comes from the 'belongs\_to' association between an Enrollment and a Course in your model!

28. Take a look! Jane appears to be doing pretty well.

[Rails U!](#)
[About](#)
[Students](#)
[Courses](#)
[Instructors](#)
[Departments](#)

Student Details

Last name

First name

Enrollment Date

Enrollments

Hacker

Jane

2014-06-01

Course Title	Grade
The Joy of Code	A
Ego Wrangling	A

Edit

Back

As usual, styling and displaying associations for other resources is an exercise left to the student ☺  
(As homework...) work on styling these views:

- Index pages for Instructors and Departments
- A cool Home page with a panel or carousel containing your RailsU logo or photos
- The 'show' views for Instructor and Department

And, make sure we don't have any pesky integer ID numbers showing up in our views:

- Show the instructor.name in the Departments index, and change the table header to 'Administrator'
- Style the course 'show' view and include the list of students who are enrolled in a course
- Style the instructor 'show' view and include the list of departments administered by the instructor

29. Let's implement another feature in our Contoso University 'workalike'. Contoso University provides a way to filter the student list to 'Find by name'. Let's give that a try.  
First, if you haven't done so, style your student index view in a similar way used to style the course index.

Open **app/views/students/index.html.erb** and check your styling:

```
1 <p id="notice"><%= notice %></p>
2 <div class="panel panel-default">
3   <div class="panel-heading">
4     <h3 class="panel-title">Students</h3>
5   </div>
6   <div class="panel-body">
7     <table class="table">
8       <thead>
9         <tr class="row">
10          <th>Last Name</th>
11          <th>First Name</th>
12          <th>Enrollment Date</th>
13          <th colspan="3"></th>
14        </tr>
15      </thead>
16      <tbody>
17        <%= @students.each do |student| %>
18          <tr class="row">
19            <td><%= student.last_name %></td>
20            <td><%= student.first_name %></td>
21            <td><%= student.enrollment_date %></td>
22            <td><%= link_to 'Show', student, :class => "btn-primary btn-sm" %></td>
23            <td><%= link_to 'Edit', edit_student_path(student), :class => "btn-primary btn-sm" %></td>
24            <td><%= link_to 'Destroy', student, :class => "btn-warning btn-sm", method: :delete, data: { confirm: 'Are you
sure?' } %></td>
25          </tr>
26        <%= end %>
27      </tbody>
28    </table>
29  </div>
30  <div class="panel-footer"><%= link_to 'New Student', new_student_path, :class => "btn-primary btn-sm" %>
31  </div>
32 </div>
```

Now let's add a 'search' bar at the top.

30. To keep things simple, we'll just enable a search on 'last\_name', by adding a form section above the table that lists all students.

In `app/views/students/index.html.erb`, after the `'panel-body'` and before the `'table'`, insert:

```
6 <div class="panel-body">
7   <div>
8     <%= form_tag(students_path, method: "get") %>
9     <%= label_tag(:searchterm, "Find by last name:") %>
10    <%= text_field_tag(:searchterm) %>
11    <%= submit_tag("Search") %>
12  </div>
13  <table class="table">
```

This will send the parameter 'searchterm', containing the user's search term, to the student controller when the user presses the Search button.

Guess we'd better make sure the controller is ready for this! Don't forget to save your work...

31. To implement the search, open the `app/controllers/students_controller.rb` file and add one line to the `index` method:

```
6 def index
7   @students = Student.search(params[:searchterm])
8 end
```

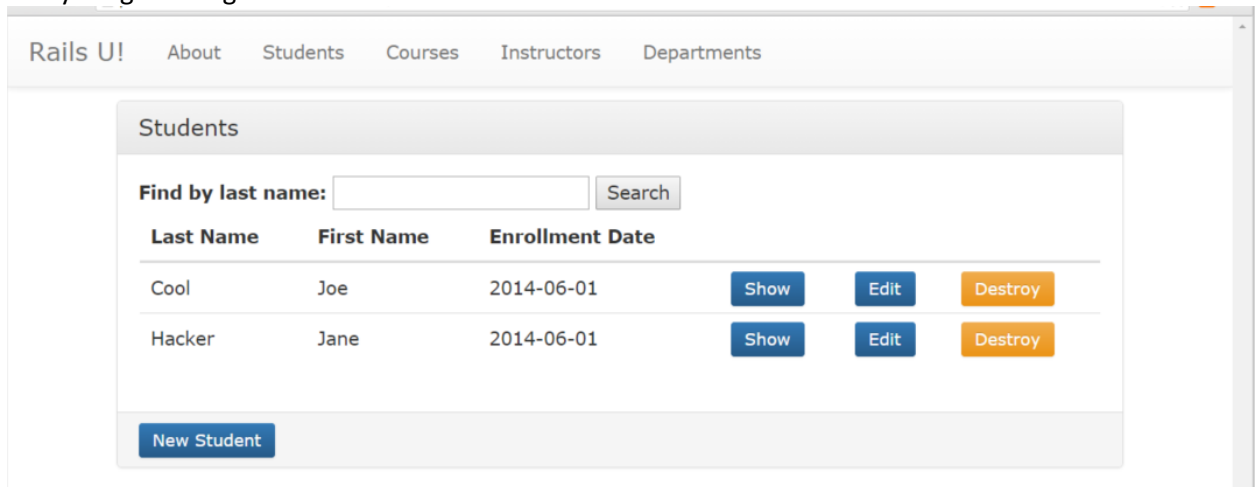
32. Ok, the controller is calling a method on the Student model called 'search' and passing it the searchterm parameter.

Guess we'd better implement that method in the Student model...

Open the `app/models/student.rb` file and add a simple search method:

```
1 class Student < ActiveRecord::Base
2   has_many :enrollments
3
4   def self.search(searchterm)
5
6     if !searchterm
7       Student.all
8     elsif searchterm.length == 0
9       Student.all
10    else
11      Student.where(last_name: searchterm)
12    end
13
14  end
15 end
```

33. Everything working?



34. What if we have a loooooong list of students? In that case we would want to do some pagination. Let's add that as well.

To the Gemfile!

Open your **Gemfile** (at the root of your app directory) and add:

```
gem 'faker'  
gem 'will_paginate'
```

We'll use the faker gem to generate a loooooong list of students, then use the will\_paginate gem to enable paging on the students index view.

Update your Gemfile as shown below (insert two new gems at the top), and save your work.

```
1 source 'https://rubygems.org'  
2  
3 gem 'faker'  
4 gem 'will_paginate'  
5  
6 # Bundle edge Rails instead: gem 'rails', github: 'rails/rails'  
7 gem 'rails', '4.2.0'  
8 # Use sqlite3 as the database for Active Record  
9 gem 'sqlite3'  
10 # Use SCSS for stylesheets  
11 gem 'sass-rails', '~> 5.0'  
12 # Use Uglifier as compressor for JavaScript assets
```

35. At your command prompt, make sure you are working in the workspace/rails/RailsU directory, and enter the command:

```
bundle
```

36. Once the bundle completes, use the rails console to add 20 or 30 new students, using our friend 'faker'. Enter the following commands:

**rails console**

(after you see the IRB prompt, enter the following as a single command)

```
20.times do Student.create!(first_name: Faker::Name.first_name,  
                             last_name: Faker::Name.last_name, enrollment_date: Date.new(2014,4,20))
```

**end**

**exit**

37. Check out those students! We definitely need some paging on this list.

Rails U!   About   Students   Courses   Instructors   Departments					
Students					
Find by last name:		<input type="text"/>	<input type="button" value="Search"/>		
Last Name	First Name	Enrollment Date			
Cool	Joe	2014-06-01	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Hacker	Jane	2014-06-01	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Murazik	Aileen	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Kuhn	Arjun	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Bauch	Evie	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Fahey	Marisa	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Heidenreich	Julie	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Cummings	Joanny	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Stanton	Hertha	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Hermiston	Lawson	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Collins	Brando	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Blick	Elva	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Wehner	Zoey	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Prosacco	Pearl	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Schuppe	Kendra	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
Wisokv	Annie	2014-04-20	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>

38. Now for the 'will\_paginate' gem.

If you would like to see all the different options and ways to page an Active Record list (like our students list), check out the documentation at [https://github.com/mislav/will\\_paginate/wiki](https://github.com/mislav/will_paginate/wiki). To get started, we'll add one line to the student model and set a limit on the number of students per page.

Open the **app/models/student.rb** file and add this line after the has\_many association:

**self.per\_page = 10**

```
1 class Student < ActiveRecord::Base
2   has_many :enrollments
3
4   self.per_page = 10
5
6   def self.search(searchterm)
7
```

Don't forget to save your work-

39. Since the controller has a lot to say in the resulting view, we'll add the pagination capability in our student controller.

Open the **app/controllers/student\_controller.rb** file and 'chain' the paginate method to the results of the search that returns the list of students for your index view:

```
4 # GET /students
5 # GET /students.json
6 def index
7   @students = Student.search(params[:searchterm]).paginate(:page => params[:page])
8 end
```

Pretty cool how we can chain those methods together! The single line of code in your index method should look like this:

**@students = Student.search(params[:searchterm]).paginate(:page => params[:page])**

40. Finally, we'll add the pagination links to the view.

Open the **app/views/students/index.html.erb** file and add the pagination links after the table of student records, towards the bottom of the file. Just insert this line:

**<%= will\_paginate @students %>**

as shown below:

```
32 <% end %>
33 </tbody>
34 </table>
35 <%= will_paginate @students %>
36 </div>
37 <div class="panel-footer"><%= link_to 'New Student', new_student_path, :class => "btn-primary btn-sm" %>
38 </div>
39 </div>
```

#### 41. Let's take a look!

Rails U!   About   Students   Courses   Instructors   Departments

---

### Students

Find by last name:

Last Name	First Name	Enrollment Date			
Cool	Joe	2014-06-01	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Hacker	Jane	2014-06-01	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Murazik	Aileen	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Kuhn	Arjun	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Bauch	Evie	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Fahey	Marisa	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Heidenreich	Julie	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Cummings	Joanny	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Stanton	Hertha	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Hermiston	Lawson	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>

← Previous 1 2 3 Next →

42. The paging works well with our Search bar. Be sure to test your app and try out the paging and search capabilities!

43. But, the paging links aren't very pretty. Let's add some style. The `will_paginate` gem documentation has links to some CSS that we can use. let's copy one of their examples into our app.

Open the **app/assets/stylesheets/scaffold.scss** file and paste in the example for digg.com styles. The CSS sample can be found here:

[http://mislav.uniqpath.com/will\\_paginate/pagination.css](http://mislav.uniqpath.com/will_paginate/pagination.css)

Or, feel free to grab this from my git repo.

44. Once you've included the style definitions in `scaffold.scss`, let's add styling to the view. Open the **app/views/student/index.html.erb** file and paste in the example for digg.com: This code should **replace** the line:

```
<%= will_paginate @students %>
```

```
<div class="digg_pagination">
  <%= will_paginate @students, :container => false %>
</div>
```

Feel free to try some of the other display options as well, or a different style (like Apple.com)

[http://mislav.uniqpath.com/will\\_paginate/?page=2](http://mislav.uniqpath.com/will_paginate/?page=2)



45. How does it look?

Rails U!

About

Students

Courses

Instructors

Departments

Students

Find by last name:

Last Name	First Name	Enrollment Date			
Collins	Brando	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Blick	Elva	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Wehner	Zoey	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Prosacco	Pearl	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Schuppe	Kendra	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Wisoky	Annie	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Hermann	Rebekah	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Hayes	Ola	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Sauer	Antonieta	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>
Monahan	Marcus	2014-04-20	<input type="button" value="Show"/>	<input type="button" value="Edit"/>	<input type="button" value="Destroy"/>

Congrats! You've passed Rails University with flying colors!