

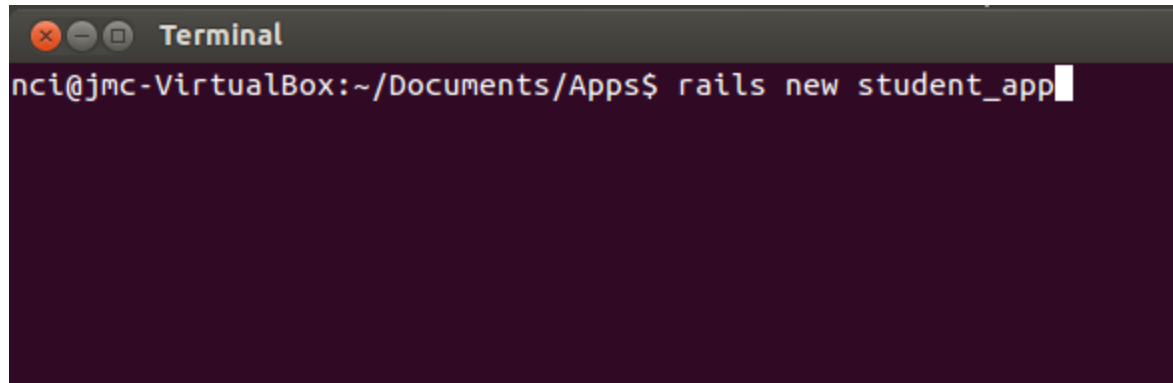
# AngularJS

CRUD Application example with AngularJS and Rails 4

1

# Create a new Rails App

- For this example we will create an application to store student details.
- Create a new rails 4 app
  - rails new student\_app



```
Terminal
nci@jmc-VirtualBox:~/Documents/Apps$ rails new student_app
```

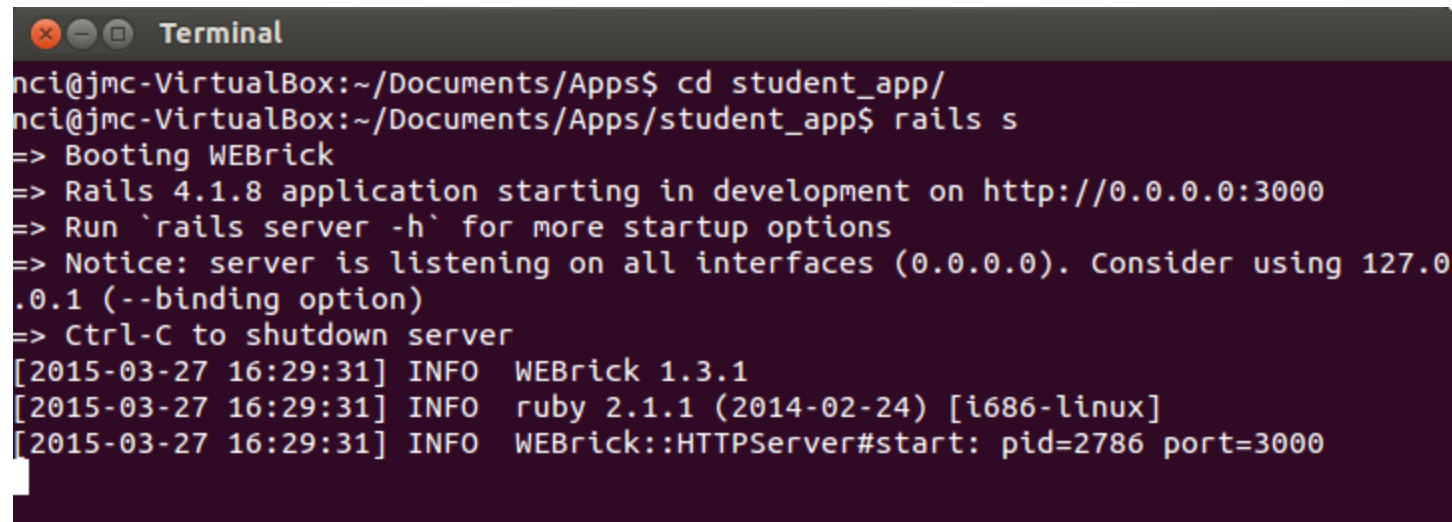
# Run the Application

- After creating the new application, cdchange directory to the new project folder.

- `cd student_app`

- Run the app

- `rails s`



```
Terminal
nci@jmc-VirtualBox:~/Documents/Apps$ cd student_app/
nci@jmc-VirtualBox:~/Documents/Apps/student_app$ rails s
=> Booting WEBrick
=> Rails 4.1.8 application starting in development on http://0.0.0.0:3000
=> Run `rails server -h` for more startup options
=> Notice: server is listening on all interfaces (0.0.0.0). Consider using 127.0.0.1 (--binding option)
=> Ctrl-C to shutdown server
[2015-03-27 16:29:31] INFO WEBrick 1.3.1
[2015-03-27 16:29:31] INFO ruby 2.1.1 (2014-02-24) [i686-linux]
[2015-03-27 16:29:31] INFO WEBrick::HTTPServer#start: pid=2786 port=3000
```

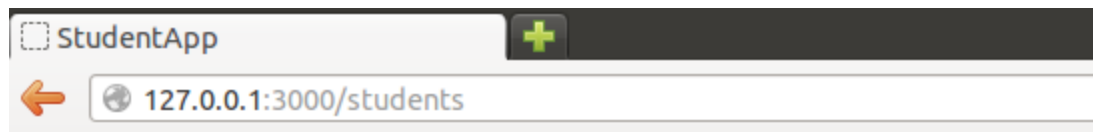
- Go to `http://127.0.0.1:3000/`

# Create a Student Scaffold

- A student should have the following values:
  - Firstname
  - Lastname
  - Student Number
  - Email Address
- Run the following command to create the scaffold
- `rails generate scaffold student firstname:string lastname:string student_num:string email:string`
- Create the database (run the following command)
  - `rake db:migrate`

# Add some sample data

- Run the server (rails s), go to <http://127.0.0.1:3000/students>
- Add 5 students to the system



## Listing students

Firstname	Lastname	Student num	Email	
Johnny	McCarthy	1574374	jon@amitycode.com	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
Bobby	Jones	15754892	bjones@amitycode.com	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
Francis	Ouimet	34356742	fouimet@amitycode.com	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
Walter	Hagen	34527865	fhagen@amitycode.com	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
Bruce	Wayne	43278453	bwayne@amitycode.com	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>

[New Student](#)

# Setup Angular

# Including Angular

- Go to: <https://code.angularjs.org/1.3.14/>
- And download angular.1.3.14.zip
- Create a new folder in assets/javascripts named angularExternal.
- Extract the zip folder contents to angularExternal

# Including jQuery and Angular

- Update the application.html.erb template to look include the following:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>  
<script src="/assets/angularExternal/angular.min.js"></script>  
<script src="/assets/angularExternal/angular-route.js"></script>  
<script src="/assets/angularExternal/angular-resource.min.js"></script>  
<script src="/assets/angular/controllers/student_controller.js"></script>
```

It should look like: (next Slide)

# Including jQuery and Angular

```
application.html.erb ✕ application.js ✕
<!DOCTYPE html>
<html ng-app="mycrudapp">
<head>
  <title>TestAngularCrud</title>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
  <script src="/assets/angularExternal/angular.min.js"></script>
  <script src="/assets/angularExternal/angular-route.js"></script>
  <script src="/assets/angularExternal/angular-resource.min.js"></script>
  <script src="/assets/angular/controllers/student_controller.js"></script>

</head>
<body>

<div class="maincontent" ng-view>
  <%= yield %>
</div>

</body>
</html>
```

# Including Angular

- By adding the two gems in the previous step, this will add Angular to the web application.
- Update the template:
- `ng-app="mycrudapp"`
- `<div class="maincontent" ng-view>`
- `<%= yield %>`
- `</div>`

# Application.html.erb

- The template file should look as follows:

```
application.html.erb x application.js x
<!DOCTYPE html>
<html ng-app="mycrudapp">
<head>
  <title>TestAngularCrud</title>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
  <script src="/assets/angularExternal/angular.min.js"></script>
  <script src="/assets/angularExternal/angular-route.js"></script>
  <script src="/assets/angularExternal/angular-resource.min.js"></script>
  <script src="/assets/angular/controllers/student_controller.js"></script>

</head>
<body>

<div class="maincontent" ng-view>
  <%= yield %>
</div>

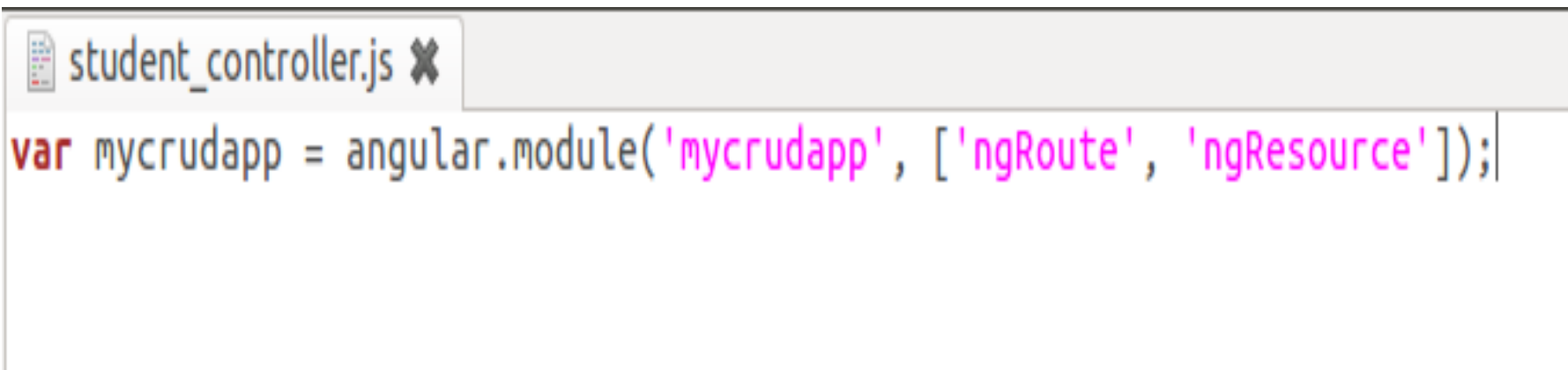
</body>
</html>
```

# Setup JS Angular files

- In the assets\Javascripts directory, create a new folder named angular
  - mkdir angular
  - cd angular
  - mkdir controllers
- This will be used to contain the logic for the Angular functionality.

# Create a Angular Student Controller

- Create a new file named **student\_controller.js** in the `/assets/javascripts/angular/controllers` folder.
- Add the following to the file:
  - `var mycrudapp = angular.module('mycrudapp', ['ngRoute', 'ngResource']);`



```
student_controller.js ✕  
var mycrudapp = angular.module('mycrudapp', ['ngRoute', 'ngResource']);
```

# Factory and Providers

- The Factory is used to send and receive information from our server side application.
- The usage of the Factory will create an object and preproperties will be added to the object. When the object is passed to the controller, these preproperties that are part of the object will be available in the controller for use through the Factory.
- See: <https://docs.angularjs.org/guide/providers>

# Setup the Factory

- **Add the following to student\_controller.js**

```
mycrudapp.factory('Students',  
  ['$resource',function($resource){  
    return $resource('/students.json', {},{  
      query: { method: 'GET', isArray: true },  
      create: { method: 'POST' }  
    })  
  }]);
```

The first Factory is named Students.

We can query students or create a new student.

# Setup the Factory

- **Add the following to student\_controller.js**

```
mycrudapp.factory('Student', ['$resource', function($resource){  
  return $resource('/students/:id.json', {}, {  
    show: { method: 'GET' },  
    update: { method: 'PUT', params: {id: '@id'} },  
    delete: { method: 'DELETE', params: {id: '@id'} }  
  });  
}]);
```

- The second Factory is named Student and allows us to show students, update an existing student and to delete a student. This is following Rails restful resources.

# student\_controller.js

- The file student\_controller.js should now look as follows:

```
var mycrudapp = angular.module('mycrudapp', ['ngRoute', 'ngResource']);

// Factory
mycrudapp.factory('Students', ['$resource', function($resource){
  return $resource('/students.json', {}, {
    query: { method: 'GET', isArray: true },
    create: { method: 'POST' }
  })
}]);

mycrudapp.factory('Student', ['$resource', function($resource){
  return $resource('/students/:id.json', {}, {
    show: { method: 'GET' },
    update: { method: 'PUT', params: {id: '@id'} },
    delete: { method: 'DELETE', params: {id: '@id'} }
  });
}]);
```

# Routing in Angular

- `$routeProvider` provides routing functionality to the Angular application with minimal effort. `$routeProvider` is the provider for the `$route` service. The `$route` service allows us to easily link controllers, templates and the URL in the address bar. The `$route` service can be used preserve the back and forward navigation in the browser.

# Setup the Routes

- **Add the following to student\_controller.js**

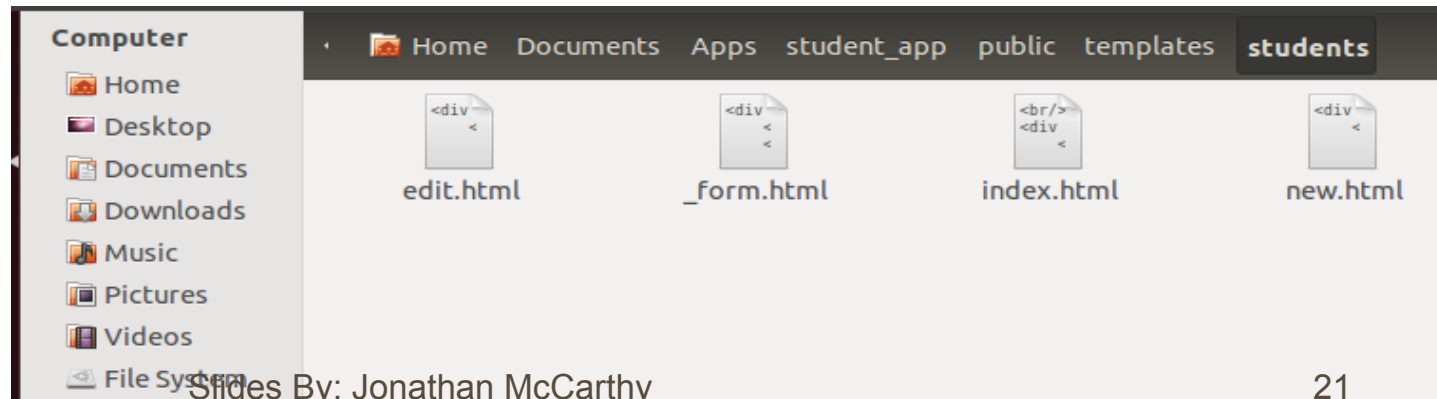
```
// This is the Routes Section
mycrudapp.config([
  '$routeProvider', '$locationProvider', function($routeProvider, $locationProvider) {
    $routeProvider.when('/students',{
      templateUrl: '/templates/students/index.html',
      controller: 'StudentListCtr'
    });
    $routeProvider.when('/students/new', {
      templateUrl: '/templates/students/new.html',
      controller: 'StudentAddCtr'
    });
    $routeProvider.when('/students/:id/edit', {
      templateUrl: '/templates/students/edit.html',
      controller: 'StudentUpdateCtr'
    });
    $routeProvider.otherwise({
      redirectTo: '/students'
    });
  }
]);
```

# Create the Angular Templates

- For this simple CRUD example, we will create the following templates:
  - Index.html
  - New.html
  - Edit.html
  - \_form.html
- The templates will be stored in the public folder in the Rails application.
- The next slide will detail the setup of the templates.

# Create the Angular Templates

- In the rails projects public folder, create a new folder named templates and in this create a folder named students. Create 4 new files in the student with the following names:
  - index.html
  - new.html
  - edit.html
  - \_form.html



# Form Explanation

- `<div class="form-group" ng-class="{ 'has-error' : submitted && studentForm.firstname.$invalid} ">`
- `<label class="control-label col-md-3">First name`  
`<span class="required">* </span> </label>`
- `<div class="col-md-4">`
- `<input type="text" name="firstname" class="form-control" ng-model="student.firstname" required placeholder="first name"/>`
- `</div>`
- `<p ng-show=" submitted && studentForm.firstname.$invalid" class="help-block">First name is required.</p>`
- `</div>`

# \_form.html

```
<div class="form-group" ng-class="{ 'has-error' : submitted && studentForm.firstname.$invalid }">
  <label class="control-label col-md-3">First name <span class="required">* </span> </label>
  <div class="col-md-4">
    <input type="text" name="firstname" class="form-control" ng-model="student.firstname" required placeholder="first name"/>
  </div>
  <p ng-show="submitted && studentForm.firstname.$invalid" class="help-block">First name is required.</p>
</div>

<div class="form-group" ng-class="{ 'has-error' : submitted && studentForm.lastname.$invalid }">
  <label class="control-label col-md-3">Last name <span class="required">* </span> </label>
  <div class="col-md-4">
    <input type="text" name="lastname" class="form-control" ng-model="student.lastname" required placeholder="last name"/>
  </div>
  <p ng-show="submitted && studentForm.lastname.$invalid" class="help-block">Last name is required.</p>
</div>

<div class="form-group" ng-class="{ 'has-error' : submitted && studentForm.email.$invalid }">
  <label class="control-label col-md-3">Email <span class="required">* </span> </label>
  <div class="col-md-4">
    <input type="email" name="email" class="form-control" ng-model="student.email" required placeholder="test@example.com"/>
  </div>
  <p ng-show="submitted && studentForm.email.$error.required" class="help-block">Email is required.</p>
  <p ng-show="submitted && studentForm.email.$error.email" class="help-block">Enter valid email</p>
</div>

<div class="form-group">
  <label class="control-label col-md-3">Student Number </label>
  <div class="col-md-4">
    <input type="text" name="student_num" class="form-control" ng-model="student.student_num" placeholder="9876543210"/>
  </div>
  <p ng-show="submitted && studentForm.student_num.$invalid" class="help-block">student Number is required.</p>
</div>

<div class="form-group">
  <div class="pull-left">
    <a class="btn btn-default" href="/"> Back</a>
    <button type="submit" class="btn btn-primary" ng-click="submitted=true">Submit</button>
  </div>
</div>
```

# new.html

```
<div class="col-md-10">  
  <form class="tab-pane active form-horizontal" id="first"  
    name="studentForm" novalidate ng-submit="save()">  
    <h3 class="block">Add new student</h3>  
    <div ng-include src="/templates/students/_form.html"></div>  
  </form>  
</div>
```

# edit.html

```
<div class="col-md-10">  
  <form class="tab-pane active form-horizontal" id="first"  
    name="studentForm" novalidate ng-submit="update()">  
    <h3 class="block">Edit student</h3>  
    <div ng-include src="/templates/students/_form.html"></div>  
  </form>  
</div>
```

# index.html

```
<br/>
<div class="row">
  <div class="col-md-12">
    <a class="btn btn-primary" href="#/students/new">Create a student</a>
    <h3 class="block">Students</h3>
    <table class="table table-striped">
      <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Student Number</th>
        <th>Email</th>
        <th></th>
      </tr>
      <tr ng-hide="students.length" >
        <td colspan="5">No students found, Please create one.</td>
      </tr>
      <tr ng-show="students.length" ng-repeat="student in students">
        <td>{{student.firstname}}</td>
        <td>{{student.lastname}}</td>
        <td>{{student.student_num}}</td>
        <td>{{student.email}}</td>
        <td>
          <a href="#/students/{{student.id}}/edit">Edit</a> | <a href="" ng-click="deleteStudent(student.id)">Remove</a>
        </td>
      </tr>
    </table>
  </div>
</div>
```

# Adding Controllers to the App

- Next we will add the Angular controllers to the application

# List Controller

```
mycrudapp.controller("StudentListCtr", ['$scope', '$http', '$resource',  
  'Students', 'Student', '$location', function($scope, $http, $resource,  
  Students, Student, $location) {
```

```
  $scope.students = Students.query();
```

```
  $scope.deleteStudent = function (studentId) {  
    if (confirm("Are you sure you want to delete this student?")){  
      Student.delete({ id: studentId }, function(){  
        $scope.students = Students.query();  
        $location.path('/');  
      });  
    }  
  };  
});
```

# Add Controller

```
mycrudapp.controller("StudentAddCtr", ['$scope', '$resource', 'Students',
  '$location', function($scope, $resource, Students, $location) {
  $scope.student = {}
  $scope.save = function () {
    if ($scope.studentForm.$valid){
      console.log('Here' + $scope.student);
      //Students.create({student: $scope.student}, function(){
      Students.create($scope.student, function(){
        $location.path('/');
      }, function(error){
        console.log(error)
      });
    }
  }
}]);
```

# Update Controller

```
mycrudapp.controller("StudentUpdateCtr", ['$scope', '$resource', 'Student',
'$location', '$routeParams', function($scope, $resource, Student, $location,
$routeParams) {
  $scope.student = Student.get({id: $routeParams.id})
  $scope.update = function(){
    if ($scope.studentForm.$valid){
console.log('Here2' + $scope.student);
      Student.update($scope.student,function(){
        $location.path('/');
      }, function(error) {
        console.log(error)
      });
    }
  };
}]);
```

# Where to place the controller code

- Add the code for the controllers to the `student_controller.js`

# CSRF – Edit application\_controller.rb

- **Update /app/controllers/application\_controller.rb to look as follows. The changes will allow Rails to work with the Angular app.**

```
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception

  after_filter :set_csrf_cookie_for_ng

  def set_csrf_cookie_for_ng
    cookies['XSRF-TOKEN'] = form_authenticity_token if protect_against_forgery?
  end

  protected

  def verified_request?
    super || form_authenticity_token == request.headers['X-XSRF-TOKEN']
  end

end
```

# Update Routes.rb

- Update routes.rb to look as follows:

```
Rails.application.routes.draw do
```

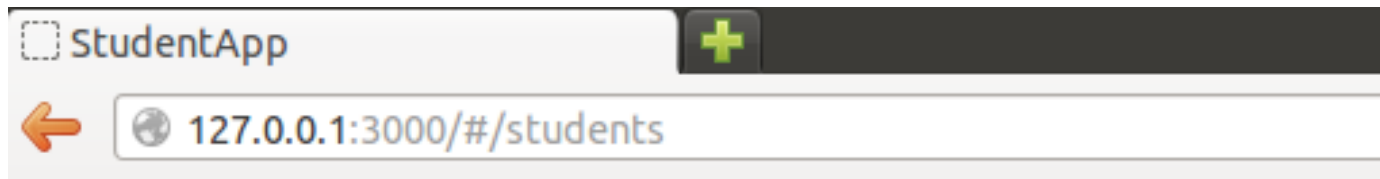
```
  resources :students
```

```
  root 'students#index'
```

```
end
```

# Run the App

- Run the application and test the functionality.
- <http://127.0.0.1:3000>



Create a student

## Students

First Name	Last Name	Student Number	Email	
Johnny	McCarthy	11111	johnny@amitycode.com	<a href="#">Edit</a>   <a href="#">Remove</a>
Bobby	Jones	15754892	bjones@amitycode.com	<a href="#">Edit</a>   <a href="#">Remove</a>
Francis	Ouimet	34356742	fouimet@amitycode.com	<a href="#">Edit</a>   <a href="#">Remove</a>
Walter	Hagen	34527865	fhagen@amitycode.com	<a href="#">Edit</a>   <a href="#">Remove</a>
Bruce F.	Wayne	43278453	bwayne@amitycode.com	<a href="#">Edit</a>   <a href="#">Remove</a>
Rocky	Balboa	12345	rocky@email.com	<a href="#">Edit</a>   <a href="#">Remove</a>

# Questions

