PMR3304 - Sistemas de Informação - 06

Aula 06

Prof. Dr. Marcos de Sales Guerra Tsuzuki

04 de Outubro de 2019

PMR-EPUSP

Desenvolvendo a Relação entre Modelos - Muitos para Muitos

Conectando Courses e Students - Introdução

Para este exemplo, vamos criar um modelo para *course*, em seguida vamos criar uma migration para **CreateCoursesStudents**. Observe que estamos continuando a partir do final da aula passada.

```
rails generate scaffold course name:string
rails generate migration CreateCoursesStudents
```

Para você fazer em casa

Execute os comandos descritos.

Conectando Courses e Students - Introdução

Vamos modificar o arquivo de migration para incluir dois campos. Após estas modificações, o comando **rails db:migrate** deve ser executado.

```
class CreateCoursesStudents < ActiveRecord::Migration[5.2]
  def change
    create_table :courses_students do |t|
      t.integer :course_id, null: false
      t.integer :student_id, null: false
    end
    add_index :courses_students, [:course_id, :student_id],
        unique: true
  end
end
```

Vamos modificar o arquivo **app/models/student.rb** de model para o *students* (uma única linha foi inserida):

```
class Student < ApplicationRecord
  # a student can have many awards
  has_many :awards, dependent: :destroy
  has_and_belongs_to_many :courses

  def name
     given_name + " " + family_name
  end
end
```

Uma modificação semelhante ocorre no arquivo **app/models/course.rb** de model para o *courses* (uma única linha foi inserida):

```
class Course < ApplicationRecord
  has_and_belongs_to_many :students
end
```

Após estas modificações, o rails utilizará a tabela courses_students recém criada para suportar estas modficações, graças ao padrão de nome.

Vamos criar um método que identifica se um *student* está matriculado em um *course*. Este método deverá ser criado no arquivo **app/models/student.rb**.

```
def enrolled_in?(course)
  self.courses.include?(course)
end
```

O método *enrolled_in?* utiliza o método *include?* de *courses* para identificar se um curso particular está incluso na lista.

Um outro método conveniente retorna uma lista de cursos que o estudante ainda não está matriculado. Este método deverá ser criado no arquivo **app/models/student.rb**.

```
def unenrolled_courses
  Course.all – self.courses
end
```

O método *Course.all* retorna a lista completa de cursos disponível. Em seguida, *self.courses* retorna a lista de cursos que o estudante está matriculado. Finalmente, a operação de subtração realiza uma operação de conjunto, removendo os elementos comuns.

Vamos acrescentar alguns métodos para simplificar as manipulações, vamos incluir no arquivo **app/controller/courses_controller.rb**

```
# GET /courses/1/roll
def roll
  @course = Course.find(params[:id])
end
```

Acrescente este método após o método *destroy* ao final do *controller*, mas antes do *private*. Este método que requer uma view **roll.html.erb**, fornecerá uma lista de estudantes que estão matriculados em um curso.

Vamos incluir um método que fornece a lista de cursos que um estudante está matriculado, no arquivo **app/controllers/students_controller.rb**.

```
# GET /courses/1/courses
def courses
  @student = Student.find(params[:id])
  @courses = @student.courses
end
```

Este método recupera o *id* a partir do roteamento e o transforma em um objeto. Neste caso, um par de objetos, representando o estudante e os cursos em que ele está matriculado.

O próximo método é um pouco diferente do que já vimos até agora. Ao invés de transferir dados para uma *view*, ele coleta informações a partir do roteamento e utililiza estas informações para modificar e direcionar o usuário para uma página convencional. Este método deve ser incluído no arquivo **app/controllers/student_controller.rb**.

Conectando Courses e Students - Modificando *Controllers*

```
# POST /students/1/course_add?course_id=2
def course_add
  # Convert ids from routing to objects
  @student = Student.find(params[:id])
  @course = Course.find(params[:course])
  unless @student.enrolled_in?(@course)
    @student.courses << @course
    flash[:notice] = 'Student was successfully enrolled'
  else
    flash[:error] = 'Student was already enrolled'
  end
  redirect_to action: "courses", id: @student
end
```

O método *course_add* utiliza o método *enrolled_in?* definido anteriormente no modelo para verificar se o estudante está matriculado na disciplina. Em caso contrário, a disciplina é acrescentada à lista de disciplinas do aluno.

A seguir será apresentado o método *course_remove*. Este método também deve ser incluído no arquivo **app/controllers/student_controller.rb**.

```ruby
# POST /student/1/course_remove?courses[]=
def course_remove
  # Convert ids from routing to objects
  @student = Student.find(params[:id])
  course_ids = params[:courses]
  if course_ids.any?
    course_ids.each do |course_id|
      course = Course.find(course_id)
      if @student.enrolled_in?(course)
        logger.info "Removing student from course #{course.id}"
        @student.courses.delete(course)
        flash[:notice] = 'Course was successfully deleted'
```

```
    if course_ids.any?
      course_ids.each do |course_id|
        course = Course.find(course_id)
        if @student.enrolled_in?(course)
          logger.info "Removing student from course #{course.id}"
          @student.courses.delete(course)
          flash[:notice] = Course was successfully deleted
        end
      end
    end
    redirect_to action: "courses", id: @student
  end
```

Conectando Courses e Students - Modificando *Routing*

É necessário modificar o roteamento, arquivo **config/routes.rb**.

```
Rails.application.routes.draw do
  resources :courses do
    member do
      get :roll
    end
  end
  resources :students do
    resources :awards
    member do
      get :courses
      post :course_add
      post :course_remove
    end
  end
end
```

Para cimentar a relação entre estudantes e disciplinas requer fornecer acesso ao usuário para as funcionalidades criadas nos *controllers* e *models*. As *views* criadas pelos *scaffolding* fornece acesso básico tanto à *students* como à *courses*. Mas não existe uma conexão entre os dois. Um primeiro passo será criar links para permitir a movimentação de um para o outro. Vamos criar o arquivo **app/views/application/_navigation.html.erb**, veja que o diretório **application** precisará ser criado.

Conectando Courses e Students - Suportanto a Relação por *Views*

O conteúdo é simples:

```
<p>
  <%= link_to "Students", students_path %> |
  <%= link_to "Courses", courses_path %>
</p>

<hr>
```

Este conteúdo pode ser referenciado a partir de todas as *views*, mas isto é um tanto incoveniente. Vamos inclui-lo no arquivo **app/views/layouts/application.html.erb**.

```
<body>
  <%= render 'navigation' %>
  <%= yield %>
</body>
```

Para você fazer em casa

Teste no browser para verificar o efeito da linha:

`localhost:3000/students/1`

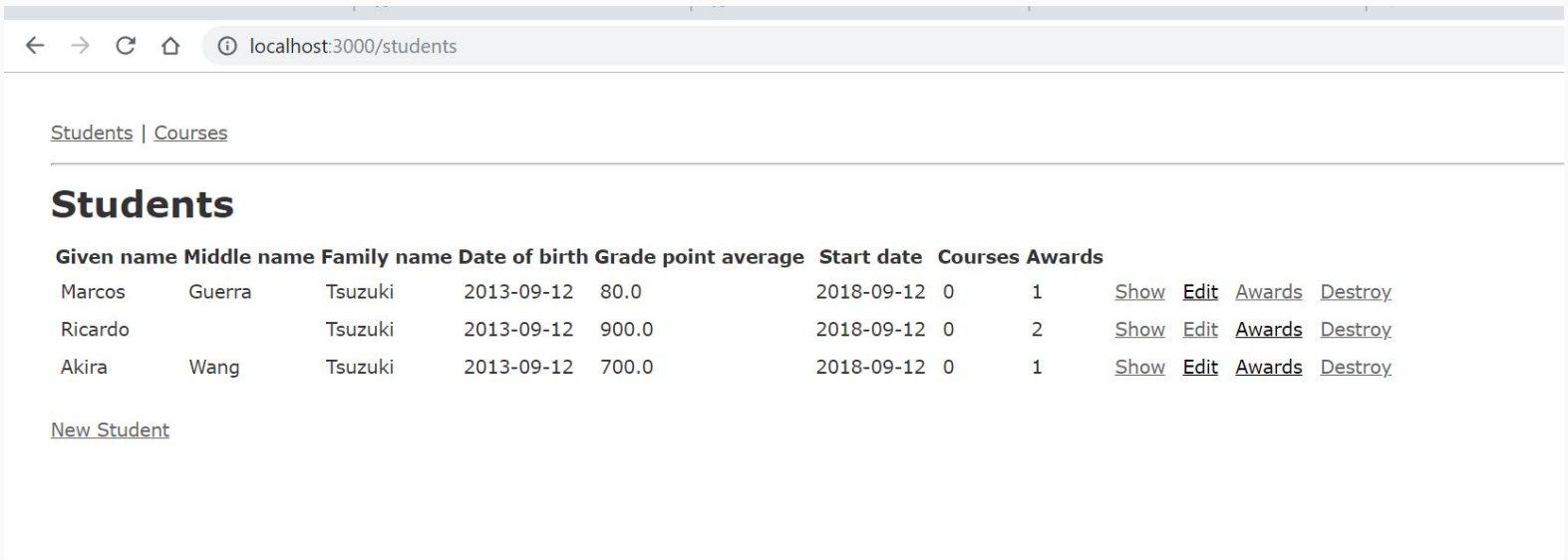
Na index page para *students* **app/views/students/index.html.erb** temos a exibição de um contador para *awards*. Vamos incluir um contador para *courses* também. Vamos incluir um título imediatamente antes de Awards:

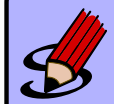
```
<th>Start date</th>
<th>Courses</th>
```

Em seguida, vamos inserir o item imediatamente antes da contagem de *awards*.

```
<td><%= student.start_date %></td>
<td><%= student.courses.count %></td>
```

Conectando Courses e Students - Suportando a Relação por *Views*





Para você fazer em casa

Teste no browser para verificar o efeito da linha:

`localhost:3000/students`

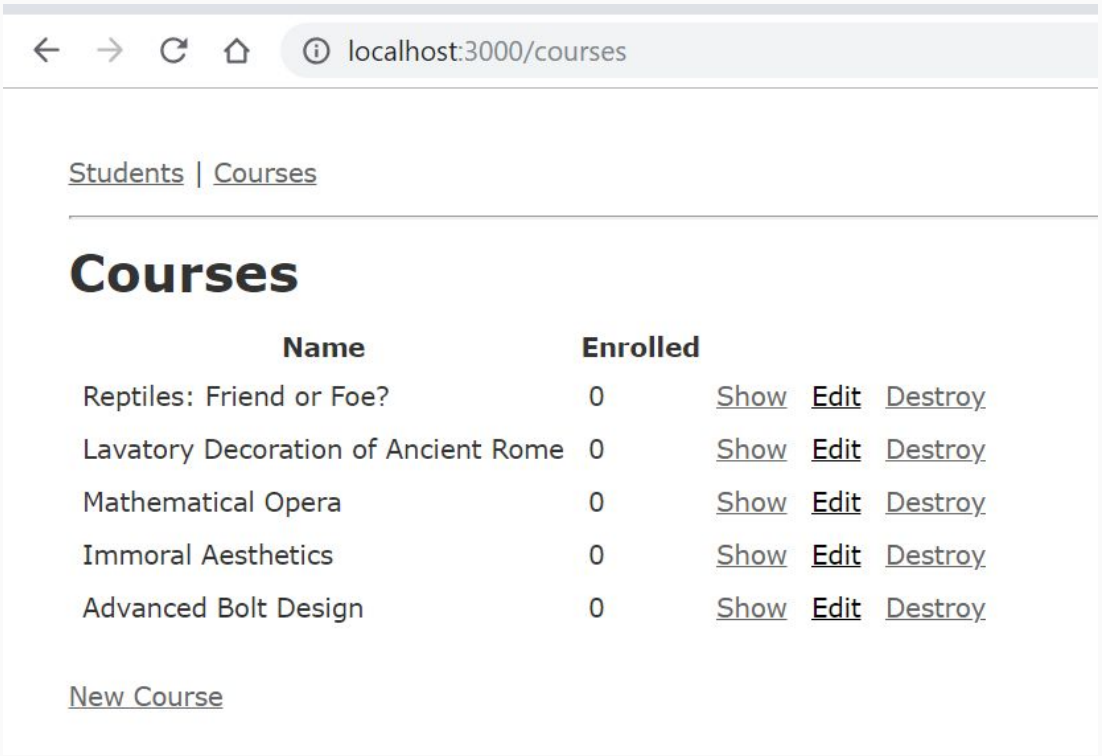
Na index page para *courses* **app/views/courses/index.html.erb** vamos incluir um contador para *students* também. Vamos incluir um título imediatamente antes de Enrolled:

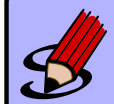
```
<th>Name</th>
<th>Enrolled</td>
<th colspan="3"></th>
```

Em seguida, vamos inserir o item de contagem.

```
<td><%= course.name %></td>
<td><%= course.students.count %></td>
<td><%= link_to 'Show', course %></td>
```

Conectando Courses e Students - Suportando a Relação por *Views*





Para você fazer em casa

Teste no browser para verificar o efeito da linha:

`localhost:3000/courses`

Conectando Courses e Students - Inscrevendo Estudantes em Cursos

Para cada estudante vamos exibir os cursos inscritos, assim, no arquivo **app/views/students/show.html.erb** vamos incluir o código (imediatamente antes de *awards*):

```
<p>
  <strong>Courses:</strong>
  <% if !@student.courses.empty? %>
    <%= @student.courses.collect {|c| link_to(c.name, c)}.join("
        , ").html_safe %>
  <% else %>
    Not enrolled in any courses yet.
  <% end %>
</p>
```

No mesmo arquivo vamos acertar os links ao final da página:

```
<%= link_to 'Edit', edit_student_path(@student) %> |
<%= link_to 'Courses', courses_student_path(@student) %> |
<%= link_to 'Awards', student_awards_path(@student) %> |
<%= link_to 'Back', students_path %>
```

Como um novo link foi incluído, o atributo de *colspan* precisa ser modificado de 3 para 4, no arquivo **app/views/students/index.html.erb**.

```
<th>Courses</th>
<th>Awards</th>
<th colspan="4"></th>
```

Ao selecionar o link *courses_url* será acionado *courses_student_path* com um estudante específico. Será acionada uma página `http://localhost:3000/students/1/courses` que ainda não foi definida. Vamos criar o arquivo **app/views/students/courses.html.erb**.

```
<h1><%= @student.name %>'s courses</h1>

<% if @courses.length > 0 %>
        <%= form_tag(course_remove_student_path(@student)) do %>
```

Conectando Courses e Students - Inscrevendo Estudantes em Cursos

```
<table> <thead> <tr>
        <th>Course</th><th>Remove?</th>
        </tr></thead>
  <tbody>
        <% for course in @courses do %>
        <tr><td><%= course.name %></td>
        <td><%= check_box_tag "courses[]", course.id %></td>
        </tr><% end %></tbody>
</table>
<br />
<%= submit_tag 'Remove checked courses' %>
<% end %>
<% else %>
        <p>Not enrolled in any courses yet.</p>
<% end %>
```

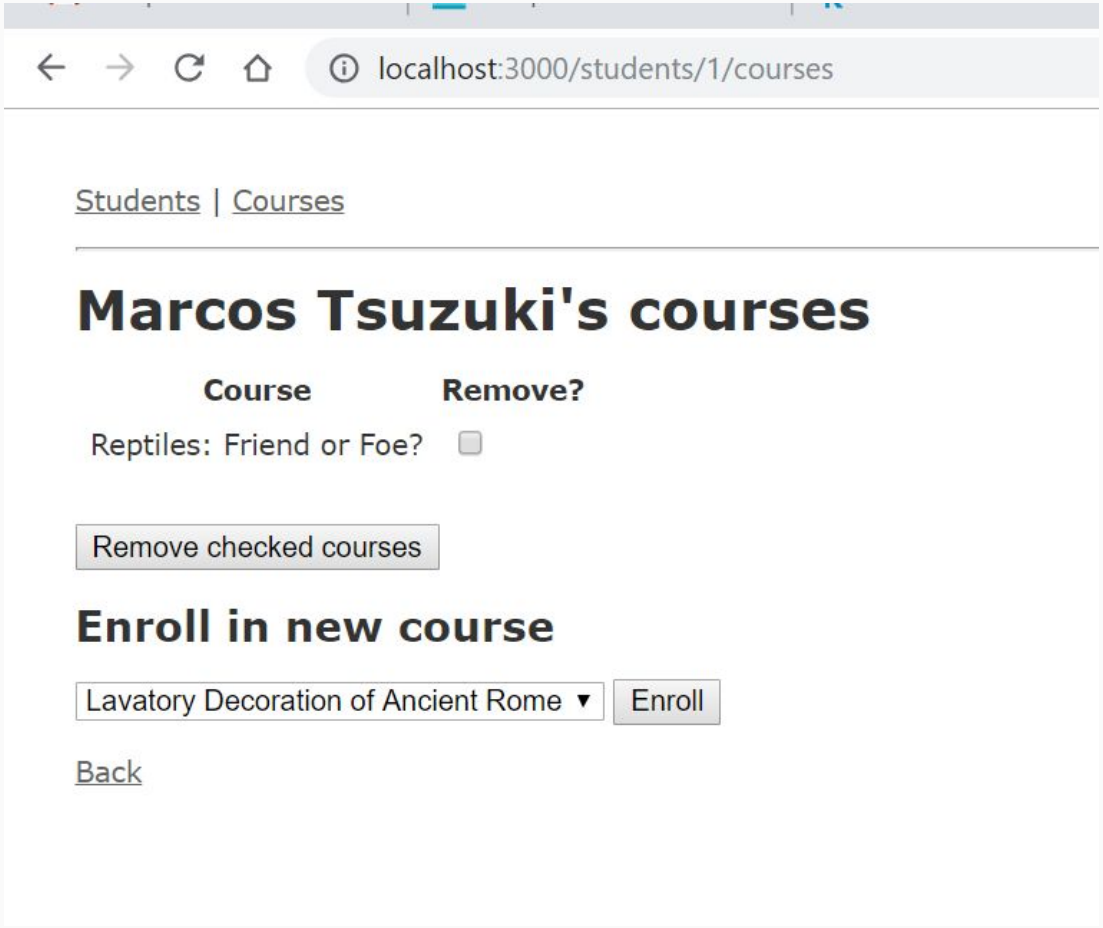
Conectando Courses e Students - Inscrevendo Estudantes em Cursos

```
<h2>Enroll in new course</h2>
<% if @student.courses.count < Course.count then %>
        <%= form_tag(course_add_student_path(@student)) do %>
        <%= select_tag(:course, options_from_collection_for_select(
              @student.unenrolled_courses, :id, :name)) %>
        <%= submit_tag 'Enroll' %>
        <% end %>
<% else %>
        <p><%= @student.name %> is enrolled in every course.</p>
<% end %>
<p><%=link_to "Back", @student %></p>
```

Esta *view* contém dois forms. Ele é criado utilizando *form_tag* ao invés de *form_for* pois ele não está associado a um *model*. O primeiro form aparece caso o estudante esteja matriculado em algum curso, permitindo que o curso seja removido. O segundo form aparece se existirem cursos que o estudante não esteja matriculado. Cada um destes forms estão conectados a métodos do *controller* para estudantes: o primeiro com *course_remove* e o segundo para *course_add*.

O form para remover cursos utiliza uma lista de *checkboxes* criados pela lista de cursos. E o form para acrescentar cursos utiliza o método *options_from_collection_for_select* que recebe uma lista de cursos de *@student.unenrolled_courses* e dois valores. O primeiro, *:id*, é o valor para ser devolvido se a linha do combo box for selecionada, e o segundo, *:name*, é o valor que será exibido no form.

Conectando Courses e Students - Suportanto a Relação por *Views*



Para você fazer em casa

Teste no browser o código desenvolvido.

Falta criar uma lista de matricula para os alunos que estão registrados em uma disciplina. No arquivo **app/views/courses/show.html.erb** vamos incluir um link:

```
<%= link_to 'Edit', edit_course_path(@course) %> |
<%= link_to 'Roll', roll_course_path(@course) %> |
<%= link_to 'Back', courses_path %>
```

Vamos criar o arquivo **app/views/courses/roll.html.erb** que exibirá a lista de matricula.

Conectando Courses e Students - Suportanto a Relação por *Views*

```
<h1>Roll for <%= @course.name %></h1>
<% if @course.students.count > 0 %>
  <table><thead><tr>
    <th>Student</th>
    <th>GPA</th>
  </tr></thead><tbody>
    <% for student in @course.students do %>
    <tr><td><%= link_to student.name, student %></td>
        <td><%= student.grade_point_average %></td></tr>
    <% end %></tbody>
  </table>
<% else %>
  <p>No students are enrolled.</p>
<% end %>
<p><%= link_to "Back", @course %></p>
```

The End!