Worksheet 8: Connecting models

CSC 340

In this worksheet, we will create two models and connect them using **a one-to-many** relationship.

1. Let’s create a model called Tree, this model has two fields: string name, string category.
2. Create a second model called Apple. This model has two fields: string color, string name and connect it to the Tree model by adding a tree\_id column using: (tree:references).

rails generate scaffold (or model) Apple color:string name:string tree:references

* **tree:references** creates a reference between the Apple and the Tree model. It also automatically adds a column to the apples table called tree\_id and adds the line: **belongs\_to tree** in the Apple model.

1. Open the Tree model (the tree.rb file) and add the line: has\_many :apples
2. Apply the migrations – you can apply both at the same time.
3. Switch to the rails console, create objects of both models Tree and Apple and navigate between the two:
   1. t= Tree.create(name: “some tree”, category: “oak”)
   2. a=Apple.create(color: “color”, name: “somename”, tree\_id: 1)
   3. create more objects of both
4. Test the one-to-many relationship. Take one Tree object and list all its apples: **t.apples**
5. Take an apple object and list the Tree object it belongs to: **a.tree**

Worksheet 9: Connecting models – part 2

CSC 340

In this worksheet, we will create two models and connect them using **a many-to-many** relationship.

1. Let’s create a model called Course, this model has one field: string name.
2. Create a second model called Student. This model has one field: string name.
3. Open the model files (the course.rb and student.rb files) and add the line:

has\_and\_belongs\_to\_many :students #added to the course.rb file

Has\_and\_belongs\_to\_many :courses #added to the student.rb file

1. Apply the migrations – you can apply both at the same time.
2. Now, we need to create a table that combines both models. To do this, we will create a migration with a specific name that will join both the courses and students table and create a way to connect the two models. Make sure you list the models in correct alphabetical order (Course should be before Student).

rails generate migration CreateJoinTableCourseStudent Course Student

1. Open the migration file and uncomment the two lines:

t.index [:course\_id, :student\_id]

t.index [:student\_id, :course\_id]

1. Apply the migration.
2. Now, you can switch to the console view and connect the models, i.e.,

c1= Course.create(name: “Web”)

c2= Course.create(name: “CS1”)

c3= Course.create(name: “SeniorP”)

s= Student.create(name: “Khadija”, course\_ids: 1)

s.courses #lists all the courses that the Student object s is connected to.

c = Course.find(1) #find the course with id 1

c.students #lists all the Student objects connected to object c

s1 = Student.create(name: “Jay”, course\_ids: [1,2,3]) # this connects this Student object to three Course objects, the ones with ids: 1, 2 and 3

s1.courses

c.students

1. Now, you can adjust the views to enter the ids in the views if you wish to do so or have a drop down menu – see the modifications to both the new action and the \_*form partial. Also, make sure to allow the course*\_ids param to pass from the form to the controller in the student\_params method.
2. Creating drop down menu:

In students\_controller.rb file

def new

@options = {}

course = Course.all

if course

course.each do |c|

@options[c.name] = c.id

end

end

@student = Student.new

End

In \_form.html.erb

<div class="field">

<%= f.label :course\_ids %>

<!--%= f.text\_field :course\_ids %-->

<%= f.select :course\_ids, @options %>

<!--%= f.select :course\_ids, [['Web', 1], ['CS1', 2], ['Web', 3]] %-->

</div>

In the students\_controller.rb, modify the students\_params method to allow the course\_ids to pass to the controller:

def student\_params

params.require(:student).permit(:name, :course\_ids)

end