



## 1 Introduction

This document describes how to provide a Rails application that has several tables. It is for a web application that records the training runs undertaken by a group of people. So it has to model the people, the courses that can be run, the actual training runs and the people that took part in those runs.

The document assumes that a Rails environment has been created. This can be provided by following the commands in either the document *Rails HOW-TO: Installing Rails into Debian* or the document *Rails HOW-TO: Installing Rails into Windows*.

The document also assumes the reader has read the document *Rails HOW-TO: A simple Rails apps: phones*.

These documents are available at <http://www.oucs.ox.ac.uk/rails/howtos>.

## 2 A running database

I am wanting to record the training runs undertaken by a group of people.

So I want to model four things;

- A *course*. This is a route used for training. So I will want to store some description of the course.
- An *event*. This is the use of a particular course on a particular day.
- A *person*. This will be used to store some details about a person.
- A *runner*. This will be used to record the fact that some person took part in some event. (A better name would be *participant*.)

## 3 Creating the Rails application

Because the four people that went out on the first training run are called Aaron, Barry, Janet and Sebastian, I am going to create a Rails application called `jabs`.

```
mkdir -p /var/apps
cd /var/apps
rails -d mysql jabs
```

## 4 Configuring Rails's access to MySQL

As before, I'm going to provide a different `database.yml` file.

```
cd /var/apps/jabs/config
cat database.yml
cat >database.yml <<%
development:
  adapter: mysql
  port: 8116
  encoding: utf8
  database: jabs_development
  username: ruby
  password: PW4ruby
# Warning: The database defined as 'test' will be erased and
# re-generated from your development database when you run 'rake'.
# Do not set this db to the same as development or production.
test:
  adapter: mysql
  port: 8116
  encoding: utf8
  database: jabs_test
  username: ruby
```

```
password: PW4ruby
production:
  adapter: mysql
  port: 8116
  encoding: utf8
  database: jabs_production
  username: ruby
  password: PW4ruby
%
```

## 5 Configuring permissions in MySQL

As before, we also need to inform MySQL.

```
mysql -P 8116 -u root -p mysql <<%
grant all privileges on jabs_development.* \
  to ruby@localhost identified by 'PW4ruby';
grant all privileges on jabs_production.* \
  to ruby@localhost identified by 'PW4ruby';
grant all privileges on jabs_test.* \
  to ruby@localhost identified by 'PW4ruby';
flush privileges;
\q
%
```

```
PW4root
```

## 6 Getting rails to create the databases

We now get Rails to create the databases.

```
cd /var/apps/jabs
rake db:create:all
```

## 7 Dealing with courses

### 7.1 Modelling a course

I need to store details of the routes that are used for training. Although we could describe things like hillyness, terrain, ... , I will just record:

- gmap: a URL of a web page at <http://www.gmap-pedometer.com>;
- length: the length of the course (in miles);
- route\_summary: a short description of the route;
- route\_description: a longer description of the route.

### 7.2 Creating the scaffold for courses

We can now get Rails to create a scaffold for courses.

```
cd /var/apps/jabs
ruby script/generate scaffold Course \
  gmap:string \
  length:decimal \
  route_summary:string \
  route_description:text
```

## 7.3 Migrating the database for courses

I will change the migration file that Rails has created so that the length is stored with two places after the decimal point and two before.

```
cd /var/apps/jabs/db/migrate
cat *_create_courses.rb
cat >*_create_courses.rb <<%
class CreateCourses < ActiveRecord::Migration
  def self.up
    create_table :courses do |t|
      t.string :gpmmap
      t.decimal :length, :precision => 4, :scale => 2
      t.string :route_summary
      t.text :route_description
      t.timestamps
    end
  end
  def self.down
    drop_table :courses
  end
end
end
%
cd /var/apps/jabs
rake db:migrate
mysql -P 8116 -u root -p jabs_development <<%
show create table courses;
select * from courses;
\q
%
PW4root
# I got no output from the select because the table exists but is empty
```

## 7.4 Testing the scaffold for courses

We can now use the scaffold to create courses, to edit them, to delete them, ... .

```
cd /var/apps/jabs
ruby script/server -p 8119 &
ps -ef | grep ruby
http://www.abcd.ox.ac.uk:8119/courses
```

# 8 Dealing with events

## 8.1 Modelling a event

An event is used to record the fact that we used a particular course on some day. So I want to record:

- course\_id: the integer that is the number of the course;
- date: the date when we did the course;
- minutes, seconds: how long we took to do the course;
- forwards: whether we ran the course as described or whether we did it in reverse.

## 8.2 Creating the scaffold for events

Now get Rails to create a scaffold for events.

```
ruby script/generate scaffold Event \
  date:date \
  minutes:integer \
  seconds:integer \
```

```
forwards:boolean \  
course_id:integer
```

### 8.3 Migrating the database for events

In database parlance, the `course_id` field is a *foreign key*, i.e., the `course_id` field of the `events` table contains the value of an `id` field in the `courses` table.

The way in which foreign keys are established varies from one database software to another. Unfortunately, Rails does not hide this from the software developer. So we have to add a MySQL command to the migration file in order to get the MySQL server to establish the foreign key.

This is the purpose of the call of the `execute` method that is given in the body of the `up` method.

```
cd /var/apps/jabs/db/migrate  
cat *_create_events.rb  
cat >*_create_events.rb <<%  
class CreateEvents < ActiveRecord::Migration  
  def self.up  
    create_table :events do |t|  
      t.date :date  
      t.integer :minutes  
      t.integer :seconds  
      t.boolean :forwards  
      t.integer :course_id, :null => false  
      t.timestamps  
    end  
    execute "alter table events add constraint fk_event_courses  
           foreign key (course_id) references courses(id) "  
  end  
  def self.down  
    drop_table :events  
  end  
end  
%
```

We also have to alter the models for `Course` and `Event` to record more details of the relationship between these two tables. The changes are recording a *one-to-many relationship*.

```
cd /var/apps/jabs  
cat app/models/course.rb  
cat >app/models/course.rb <<%  
class Course < ActiveRecord::Base  
  has_many :events  
end  
%  
cat app/models/event.rb  
cat >app/models/event.rb <<%  
class Event < ActiveRecord::Base  
  belongs_to :course  
end  
%
```

Having made those changes, now do the migration.

```
cd /var/apps/jabs  
rake db:migrate  
mysql -P 8116 -u root -p jabs_development <<%  
show create table events;  
select * from events;  
\q
```

```
%
PW4root
# This showed the foreign key but there was no output
# from the select because the table exists but is empty
```

## 8.4 Testing the scaffold for events

We can now use the scaffold for events to do tasks such as creating new events, editing events, and deleting events.

```
ps -ef | grep ruby
kill -KILL PPP
cd /var/apps/jabs
ruby script/server -p 8119 &
ps -ef | grep ruby
http://www.abcd.ox.ac.uk:8119/events
```

As each event is connected to a particular course, the biggest problem in using the interface of the scaffold is knowing the integer values associated with each course.

This is particularly a problem when adding a new event. The form currently uses:

```
f.text_field :course_id
```

and it expects us to type an integer into this textbox.

We can get it to generate the appropriate integer automatically by using a drop-down list. This can be achieved by altering the form so that instead of:

```
f.text_field :course_id
```

it uses:

```
f.select(:course_id, @courses)
```

The `@courses` can be any object that supports the `Enumerable` mixin. So it could be an array, a hash or the result of the use of a database `find` method.

In the controller for an `Event`, we will create an array where each element is an array containing two values: the string to be displayed in the drop-down list and the associated `course_id`.

```
ed app/controllers/events_controller.rb <<%
/def new/a
  @courses =
    Course.find(:all,
      :order => "length desc, route_summary").map do |c|
      [c.length.to_s + " " + c.route_summary, c.id]
    end
.
w
q
%
ed app/views/events/new.html.erb <<'%'
g/f.text_field :course_id/s//f.select(:course_id, @courses)/gp
w
q
%
```

```
http://www.abcd.ox.ac.uk:8119/events
```

## 8.5 Wandering across tables

We can write code that seamlessly wanders across tables:

```
<table>
<% for event in @events %>
```

```
<tr>
  <td><%=h event.date %></td>
  <td><%=h event.course.length %></td>
</tr>
<% end %>
</table>
```

## 9 Dealing with people

### 9.1 Modelling a person

For each person, obviously, I will want to record first name and last name. However, I also want to record a one-letter code that can be used to refer to the person.

### 9.2 Creating the scaffold for people

Now get Rails to create a scaffold for people.

```
ruby script/generate scaffold Person \
  code:string \
  firstname:string \
  lastname:string
```

### 9.3 Migrating the database for people

We have seen that when we ask Rails to model a `Course` it creates a table called `courses`. Here we are asking Rails to model a `Person` and it automatically creates a table called `people`. Bizarrely, Rails knows about some of the weird pluralizations of the English language.

```
cd /var/apps/jabs/db/migrate
cat *_create_people.rb
cd /var/apps/jabs
rake db:migrate
mysql -P 8116 -u root -p jabs_development <<%
show create table people;
select * from people;
\q
%
PW4root
# This showed no output from select 'cos table exists but is empty
```

### 9.4 Testing the scaffold for people

We can now test the scaffold that Rails has created.

```
ps -ef | grep ruby
kill -KILL PPP
cd /var/apps/jabs
ruby script/server -p 8119 &
ps -ef | grep ruby
http://www.abcd.ox.ac.uk:8119/people
```

## 10 Dealing with runners

### 10.1 Modelling a runner

We are wanting to model who took part in a particular event. For this we will use a model that records the number of the person and the number of the event.

## 10.2 Creating the scaffold for runners

We now get Rails to create the scaffold for runners.

```
ruby script/generate scaffold Runner \  
  event_id:integer \  
  person_id:integer
```

## 10.3 Migrating the database for runners

Because both the `event_id` and `person_id` field are foreign keys, we need to add extra code to establish two one-to-many relationships.

```
cd /var/apps/jabs/db/migrate  
cat *_create_runners.rb  
cat >*_create_runners.rb <<%  
class CreateRunners < ActiveRecord::Migration  
  def self.up  
    create_table :runners do |t|  
      t.integer :event_id, :null => false  
      t.integer :person_id, :null => false  
      t.timestamps  
    end  
    execute "alter table runners add constraint fk_runner_events  
      foreign key (event_id) references events(id)"  
    execute "alter table runners add constraint fk_runner_people  
      foreign key (person_id) references people(id)"  
  end  
  def self.down  
    drop_table :runners  
  end  
end  
%  
cd /var/apps/jabs  
cat app/models/event.rb  
cat >app/models/event.rb <<%  
class Event < ActiveRecord::Base  
  belongs_to :course  
  has_many :runners  
end  
%  
cat app/models/person.rb  
cat >app/models/person.rb <<%  
class Person < ActiveRecord::Base  
  has_many :runners  
end  
%  
cat app/models/runner.rb  
cat >app/models/runner.rb <<%  
class Runner < ActiveRecord::Base  
  belongs_to :event  
  belongs_to :person  
end  
%  
cd /var/apps/jabs  
rake db:migrate  
mysql -P 8116 -u root -p jabs_development <<%  
show create table runners;  
select * from runners;  
\q  
%  
PW4root  
# This showed the foreign keys but no output
```

```
# from the select because the table exists but is empty
```

## 10.4 Testing the scaffold for runners

We can now try out the scaffold that Rails has generated. To make life easier, as before we alter the view when creating a new runner so that it presents drop-down lists in place of the event textbox and the person textbox.

```
ps -ef | grep ruby
kill -KILL PPP
cd /var/apps/jabs
ruby script/server -p 8119 &
ps -ef | grep ruby
http://www.abcd.ox.ac.uk:8119/runners
ed app/controllers/runners_controller.rb <<%
/def new/a
  @events = Event.find(:all).map do |e|
    [e.date.to_s + " " + e.course.length.to_s +
     " " + e.course.route_summary, e.id]
  end
  @people = Person.find(:all).map do |p|
    [p.firstname + " " + p.lastname, p.id]
  end
.
w
q
%
ed app/views/runners/new.html.erb <<'%'
g/f.text_field :event_id/s//f.select(:event_id, @events)/gp
g/f.text_field :person_id/s//f.select(:person_id, @people)/gp
w
q
%
http://www.abcd.ox.ac.uk:8119/runners
```