<https://www.flearning.net/lectures/9/clips/47>

rails new ex2

cd ex2

bin/rails server

rails g controller timeline

rails g model blog content:text

rails g model comment msg:text

/Users/sangeun/lucy/ex2/db/migrate/20180307084245\_create\_comments.rb

t.integer :blog\_id

(또는 처음부터rails g model comment msg:text blog\_id:integer)

rake db:migrate

(수정시, rake db:rollback 후rake db:migrate)

rails c

Blog.create(content: "first blog")

Blog.create(content: "second blog")

Blog.all

Comment.new (컬럼이 나와요)

<<마이그레이션 관련하여 ruby가이드를 보는 것도 좋을 듯>>

<https://rubykr.github.io/rails_guides/migrations.html>

Comment.create(blog\_id:1,msg:"first blog first comment")

Comment.create(blog\_id:1,msg:"first blog second comment")

/Users/sangeun/lucy/ex2/app/models/blog.rb

class Blog < ApplicationRecord

has\_many :comments

end

/Users/sangeun/lucy/ex2/app/models/comment.rb

class Comment < ApplicationRecord

belongs\_to :blog

end

rails c (레일즈 콘솔)

a=Blog.find(1)

a (검색 결과 나옴)

a.comments

b=Comment.find(1)

b

b.blog

/Users/sangeun/lucy/ex2/config/routes.rb

root 'timeline#index'

/Users/sangeun/lucy/ex2/app/controllers/timeline\_controller.rb

def index

end

<https://www.flearning.net/lectures/9/clips/48>

\*\*1강 강의 노트

**모델Model 관계Relation**

**timeline 컨트롤러 생성 명령어**

rails g controller timeline

**config/routes.rb 파일 작성하기**

root 'timeline#index'

**app/views/timeline/index.html.erb 파일 만들기**

(아래 참고 코드에 있습니다)

**Blog 모델 생성 명령어**

rails g model blog content:text

**Comment 모델 생성 명령어**

rails g model comment msg:text blog\_id:integer

위 스크립트로 생성되는 마이그레이션 파일과 동영상 강의에서 직접 마이그레이션 파일을 수정하는 것의 차이를 생각해보세요

**app/models/blog.rb**

class Blog < ActiveRecord::Base has\_many :comments end

이제 특정 Blog 글에 대한 .comments 메소드를 사용할 수 있게 되며, 해당 블로그의 comments를 호출하면 자식들이 배열에 담겨 나옵니다.

**app/models/comment.rb**

class Comment < ActiveRecord::Base belongs\_to :blog end

이제 특정 Comment 에 대해 .blog 메소드를 사용할 수 있으며, 해당 코멘트의 부모인 Blog 글을 불러옵니다.

\*\*2강강의노트

**모델Model 관계Relation**

**Comment 모델 생성 명령어**

rails g model comment msg:text blog\_id:integer

위 스크립트로 생성되는 마이그레이션 파일과 동영상 강의에서 직접 마이그레이션 파일을 수정하는 것의 차이를 생각해보세요

**app/models/blog.rb**

class Blog < ActiveRecord::Base has\_many :comments end

이제 특정 Blog 글에 대한 .comments 메소드를 사용할 수 있게 되며, 해당 블로그의 comments를 호출하면 자식들이 배열에 담겨 나옵니다.

**app/models/comment.rb**

class Comment < ActiveRecord::Base belongs\_to :blog end

이제 특정 Comment 에 대해 .blog 메소드를 사용할 수 있으며, 해당 코멘트의 부모인 Blog 글을 불러옵니다.

**레일즈 콘솔에서 테스트해봅시다.**

터미널 bash에서 콘솔을 실행합니다

rails c

콘솔에서 다음 명령어들을 한줄한줄 차례로 실행해봅시다

Blog.create(content: "first blog") Blog.create(content: "second blog") Comment.create(blog\_id: 1, msg: "1 blog 1 comment") Comment.create(blog\_id: 1, msg: "1 blog 2 comment") Comment.create(blog\_id: 2, msg: "2 blog 1 comment") first = Blog.find(1) first.comments comm = Comment.find(3) comm.blog

first.comments가 리턴하는 값을 구경해봅니다. 또, comm.blog가 리턴하는 값을 한번 살펴 봅시다.

**나머지 참고 코드**

**app/controllers/timeline\_controller.rb**

class TimelineController < ApplicationController before\_action :authenticate\_user!, except: :index def index @blogs = Blog.all.reverse end def write Blog.create(content: params[:naeyong]) redirect\_to :root end def comment Comment.create(blog\_id: params[:blog\_id], msg: params[:comment]) redirect\_to :root end end

**app/views/timeline/index.html.erb**

<h1>Timeline</h1> <%=form\_tag '/write', method: 'post' do %> <textarea name="naeyong"></textarea> <input type="submit"> <% end %> <ul> <% @blogs.each do |b| %> <li><%=b.content%> <%=form\_tag '/comment' do%> <input type="hidden" name="blog\_id" value="<%=b.id%>"> <input type="text" name="comment"> <input type="submit"> <% end %> <ol> <% b.comments.each do |c| %> <li><%=c.msg%></li> <% end %> </ol> </li> <% end %> </ul>