

Tutor . 김민정

멋쟁이 사자처럼 at 이화여대

1 : N

5 / 3 (목)



LIKE LION

Part 1 . CRUD



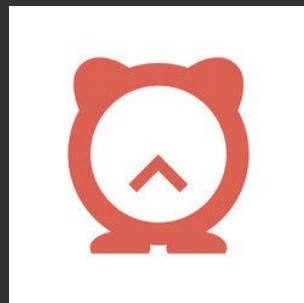
LIKE LION

엥? 게시판이 이게 끝?! 그럴리가!!



LIKE LION

우리가 생각한건!!



LIKE LION

뭐가 부족하지?

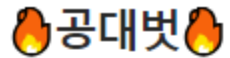


LIKE LION

Part 2 . 댓글 기능



LIKE LION



공대벗들을 위한 게시판입니다



익명

어제 10:34

신고

마지막시험까지 멋지게 조졌다ㅎㅎ

👍 22 💬 3 ☆ 0



익명 어제 10:36

대댓글

신고

ㅋㅋ나두 어쨌든 끝났다~~~~



익명 어제 10:40

신고

22맞아 어쨌거나 끝남~~~~



익명 어제 13:25

대댓글

신고

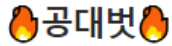
물화 슈밤~~

댓글을 입력하세요.

□ 익명



LIKE LION



공대벗들을 위한 게시판입니다



익명

04/29 23:22

신고

시발 시험이 끝나니까 팀플이랑 과제가 있네?

👍 39 💬 10 ⭐ 0



익명 04/29 23:32

대댓글

신고

딱봐도 전전



익명(글쓴이) 04/29 23:32

신고

아닌데 😊



익명(글쓴이) 04/29 23:32

신고

전전도 그럴군..



익명 04/29 23:33

대댓글

신고

내말이 진짜 개빡쳐...지금 할 거 새어보니까 10개넘어...진짜 언제 노니



익명 04/29 23:50

대댓글

신고

나는~시험도 안끝났는데~과제도 있고~팀플 발표도 있고~새 팀플도 시작됐고~룰루랄라~



익명 04/29 23:54

대댓글

신고

과제 팀플 쪽글 퀴즈... 그래도 일단 이번주말은 나물라라하고 쉬었억ㅋㅋㅋ



익명 어제 00:48

대댓글

신고

کم공인갓



익명(글쓴이) 어제 00:50

신고

벗도 کم공인가..



LIKE LION

한 개의 게시글 안에 여러 개의 댓글



LIKE LION

댓글 구현

한 개의 게시글 안에 여러 개의 댓글

1 : N



LIKE LION

CRUD

여러 개의 게시글 만들기!



LIKE LION

CRUD

1 : N

여러 개의 게시물 만들기!

한 개의 게시물에

여러 개의 댓글 만들기



LIKE LION

Model 사이의 관계 정의



LIKE LION

공대벗들을 위한 게시판입니다

익명

어제 10:34

신고

마지막시험까지 멋지게 조ť다ㅎㅎ

22

3

0

익명

어제 10:36

대답글 신고

큉큉나두 어ť트 끝났다~~~~

익명

어제 10:40

신고

22맞아 어ť트거나 끝남~~~~

익명

어제 13:25

대답글 신고

을화 슈밤~~

답글을 입력하세요.

익명

공대벗들을 위한 게시판입니다

익명

04/29 23:22

신고

시발 시험이 끝나니까 팀플이랑 과제가 있네?

39

10

0

익명

04/29 23:32

대답글 신고

딱봐도 전전

익명(글쓴이)

04/29 23:32

신고

아닌데

익명(글쓴이)

04/29 23:32

신고

전전도 그렇군..

익명

04/29 23:33

대답글 신고

내말이 진짜 개백쳐..지금 할 거 새어보니까 10개넘어..진짜 언제 노니

익명

04/29 23:50

대답글 신고

나는~시험도 안끝났는데~과제도 있고~팀플 발표도 있고~새 팀플도 시작됐고~룰루랄라~

익명

04/29 23:54

대답글 신고

과제 팀플 쪽글 퀴즈.. 그래도 일단 이번주말은 나물라라하고 쉬었억ㅋㅋ

익명

어제 00:48

대답글 신고

کم공인갓

익명(글쓴이)

어제 00:50

신고

벗도 کم공인가..

이 게시물에 CRUD를 적용하면?



CRUD - Post Model

id	content	Created_at
1	마지막 시험까지 멋지게 조졌다ㅎㅎ	2018.04.30. 10:34
2	시험이 끝나니까 팀플이랑 과제가 있네?	2018.04.29. 23:22



LIKE LION

CRUD - Post Model

id	Content	Created_at
1	마지막 시험까지 멋지게 조졌다ㅎㅎ	2018.04.30. 10:34
2	시험이 끝나니까 팀플이랑 과제가 있네?	2018.04.29. 23:22

댓글들만 따로 model로 만들자!



LIKE LION

1 : N - Comment Model

id	content	Created_at
1	ㅋㅋ 나두 어쨌든끝났다 ~~	2018.04.30. 10:36
2	22맞아 어쨌거나 끝남 ~~	2018.04.30. 10:40
3	물화 슈밤 ~~	2018.04.30. 13:25
4	내말이 진짜 개빡쳐..	2018.04.29. 23:33
5	کم공인갓	2018.04.30. 00:48
6	벗도 کم공인가	2018.04.30. 00:50
7	과제 팀플 너무 많아..	2018.04.30. 09:17
8	시험기간->과제기간->시험기간->방학	2018.04.30. 21:23



LIKE LION

Post Model

id	content	Created_at
1	마지막 시험까지 멋지게 조졌다ㅎㅎ	2018.04.30. 10:34
2	시험이 끝나니까 팀플이랑 과제가 있네?	2018.04.29. 23:22

Comment Model

id	content	Created_at
1	ㅋㅋ 나두 어쨌든끝났다 ~~	2018.04.30. 10:36
2	22맞아 어쨌거나 끝남 ~~	2018.04.30. 10:40
3	물화 슈밤 ~~	2018.04.30. 13:25
4	내말이 진짜 개빡쳐..	2018.04.29. 23:33
5	کم공인갓	2018.04.30. 00:48
6	벗도 کم공인가	2018.04.30. 00:50
7	과제 팀플 너무 많아..	2018.04.30. 09:17
8	시험기간->과제기간->시험기간->방학	2018.04.30. 21:23



LIKE LION

Post Model

id	content	Created_at
1	마지막 시험까지 멋지게 조졌다ㅎㅎ	2018.04.30. 10:34
2	시험이 끝나니까 팀플이랑 과제가 있네?	2018.04.29. 23:22

Comment Model

id	content	Created_at
1	ㅋㅋ 나두 어쨌든 끝났다 ~~	2018.04.30. 10:36
2	22맞아 어쨌거나 끝남 ~~	2018.04.30. 10:40
3	물화 슈밤 ~~	2018.04.30. 13:25
4	내말이 진짜 개박쳐..	2018.04.29. 23:33
5	کم공인갓	2018.04.30. 00:48
6	벗도 کم공인가	2018.04.30. 00:50
7	과제 팀플 너무 많아..	2018.04.30. 09:17
8	시험기간->과제기간->시험기간->방학	2018.04.30. 21:23



LIKE LION

그러면?!?!

Comment model의 1번부터 3번은 1번 게시글에!

Comment model의 4번부터 8번은 2번 게시글에!

⇒ CRUD에서 정보 불러왔던 것 처럼 불러와 버리자!!



LIKE LION

그러면?!?!

~~Comment model의 1번부터 3번은 1번 게시글에!~~

~~Comment model의 4번부터 6번은 2번 게시글에!~~

~~⇒ CRUD에서 정보 불러왔던 것 처럼 불러와 버리자!!~~



LIKE LION

Created_at 순서대로 id값이 할당!

comment model 섞인다!

게시글에 따라 정렬되지 않고 **뒤죽박죽!**



LIKE LION

Post Model

id	content	Created_at
1	마지막 시험까지 멋지게 조졌다ㅎㅎ	2018.04.30. 10:34
2	시험이 끝나니까 팀플이랑 과제가 있네?	2018.04.29. 23:22

Comment Model

id	content	Created_at
1	내말이 진짜 개박쳐..	2018.04.29. 23:33
2	کم공인갓	2018.04.30. 00:48
3	벗도 کم공인가	2018.04.30. 00:50
4	과제 팀플 너무 많아..	2018.04.30. 09:17
5	ㅋㅋ 나두 어쨌든 끝났다 ~~	2018.04.30. 10:36
6	22맞아 어쨌거나 끝남 ~~	2018.04.30. 10:40
7	물화 슈밤 ~~	2018.04.30. 13:25
8	시험기간->과제기간->시험기간->방학	2018.04.30. 21:23



LIKE LION

댓글이
어떤 게시물에
속하는 지가 중요하다!



LIKE LION

더 나아가면!

댓글이
어떤 **사용자**에
속하는 지 설정할 수 있다!



LIKE LION

id	content
1	마지막 시험까지 멋지게 조졌다ㅎㅎ

id	content
2	시험이 끝나니까 팀플이랑 과제가 있네?

id	content
5	ㅋㅋ 나두 어쨌든 끝났다 ~~
6	22맞아 어쨌거나 끝남 ~~
7	물화 슈밤 ~~

id	content
1	내말이 진짜 개빡쳐..
2	کم공인갓
3	벗도 کم공인가
4	과제 팀플 너무 많아..
8	시험기간->과제기간->시험기간->방학

게시글과 댓글 사이의 관계를 정의한다

1 : N

한 개의 게시글 안에 여러 개의 댓글



LIKE LION

Part 3 . 1 : N 실습



LIKE LION

갓

서영튜터님의 CRUD 완성판!

세션자료에서의 이름과 똑같은 원한다면! Clone!


<https://github.com/mjung1798/ruby1-n.git>






스타 주세요><



LIKE LION


This repository

Pull requests
Issues
Marketplace
Explore

mjung1798 / ruby1-n

Watch 0
Star 0
Fork 0

Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings


이화여대 멋쟁이 사자처럼 ruby 1:n세션준비

[Add topics](#)

1 commit
1 branch
0 releases
1 contributor

Branch: master
New pull request

Create new file
Upload files
Find file
Clone or download


mjung1798 first commit

app	first commit	
bin	first commit	
config	first commit	
db	first commit	
lib	first commit	6 minutes ago
log	first commit	6 minutes ago
public	first commit	6 minutes ago
test	first commit	6 minutes ago
vendor/assets	first commit	6 minutes ago
.gitignore	first commit	6 minutes ago

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

<https://github.com/mjung1798/ruby1-n.git>

Open in Desktop

Download ZIP

Use SSH



LIKE LION

Create a new workspace

Workspace name

funny_session

Description

즐거운 실습시간 >

Team

이대멋사

Hosted workspace

Clone workspace

Remote SSH workspace

Salesforce



Private

This is a workspace for your eyes only



Public

This will create a workspace for everybody to see

Clone from Git or Mercurial URL (optional)

https://github.com/mjung1798/ruby1-n.git

붙여넣기!

Choose a template



HTML5

node

Node.js

php

PHP, Apache & ...



Python

django

Django



Ruby



C++



Wordpress



Rails Tutorial



Blank



Harvard's CS50

Create workspace



LIKE LION

1) Comment 모델 만들기

```
rails g model comment content:string post_id: integer
```

- **Comment**라는 이름의 모델을 만들어낸다(generate)
- Type이 string인 **content** column
- Type이 integer인 **post_id** column



LIKE LION

1) Comment 모델 만들기

rails g model **comment** content:string post_id: integer

```
class CreateComments < ActiveRecord::Migration
  def change
    create_table :comments do |t|
      t.string :content
      t.integer :post_id

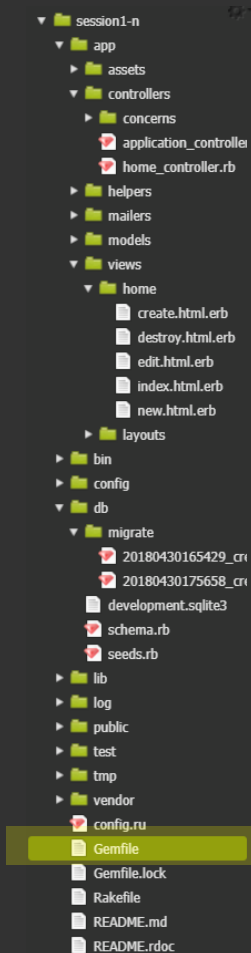
      t.timestamps null: false
    end
  end
end
```



LIKE LION

<생성된 모델 확인하기> gem rails_db

- 1) gemfile 클릭 2) gem 'rails_db' 입력



```
Gemfile
1 source 'https://rubygems.org'
2
3 gem 'rails_db'
4
5 # Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
6 gem 'rails', '4.2.5'
7 # Use sqlite3 as the database for Active Record
```

- 3) bash에 bundle install 입력

```
mjung1798:~/workspace (master) $ bundle install
Fetching gem metadata from https://rubygems.org/.....
Fetching version metadata from https://rubygems.org/..
Fetching dependency metadata from https://rubygems.org/.
Resolving dependencies...
```

* Bundle install은 서버를 끄고!



LIKE LION

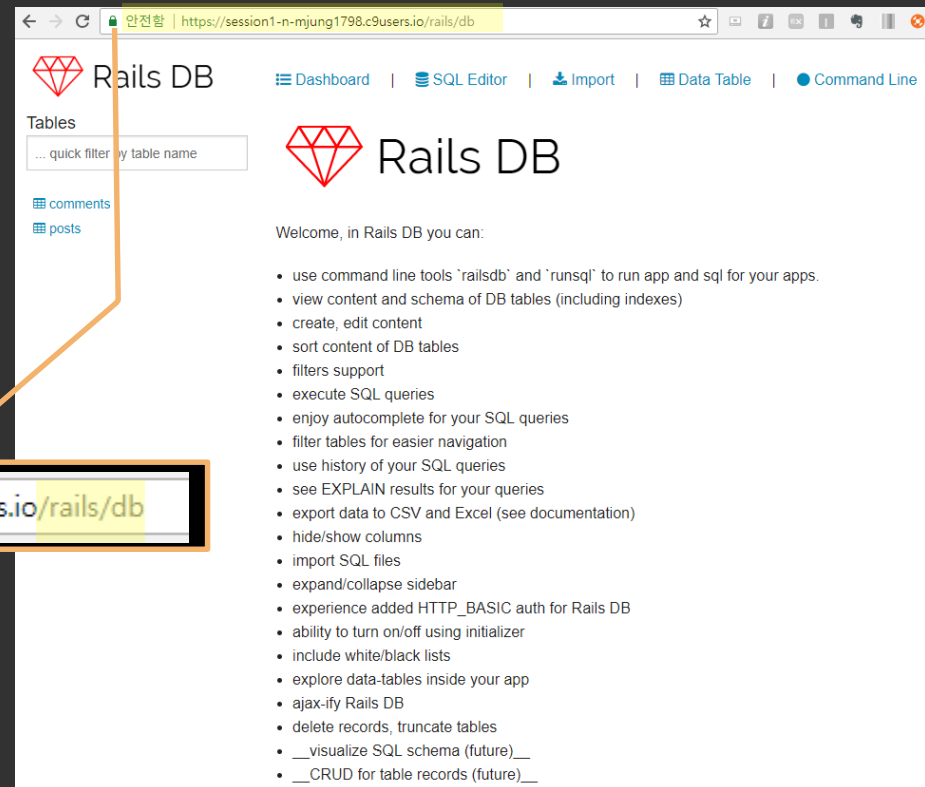
<생성된 모델 확인하기>

gem rails_db

4) Run project

rake:db 꼭 하기!

5) /rails/db url에 입력하기



안전함

https://session1-n-mjung1798.c9users.io/rails/db

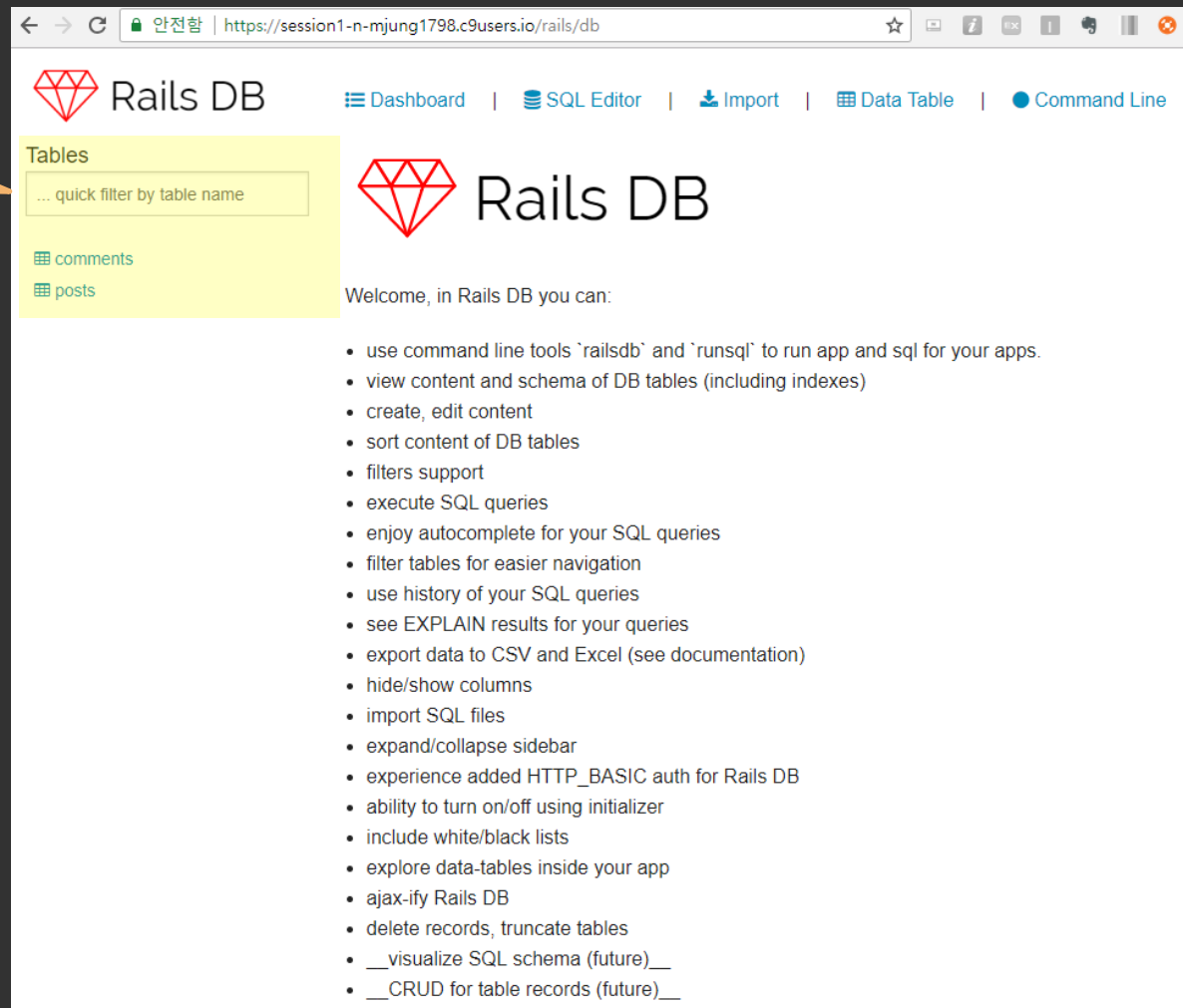


LIKE LION

<생성된 모델 확인하기>

gem rails_db

생성한 model 확인!
model = table



The screenshot shows the Rails DB web interface. The top navigation bar includes links for Dashboard, SQL Editor, Import, Data Table, and Command Line. The main content area is divided into two sections. On the left, under the heading 'Tables', there is a search bar labeled '... quick filter by table name' and a list of tables: 'comments' and 'posts'. On the right, there is a large heading 'Rails DB' with a red diamond logo. Below this, a welcome message states 'Welcome, in Rails DB you can:' followed by a bulleted list of features:

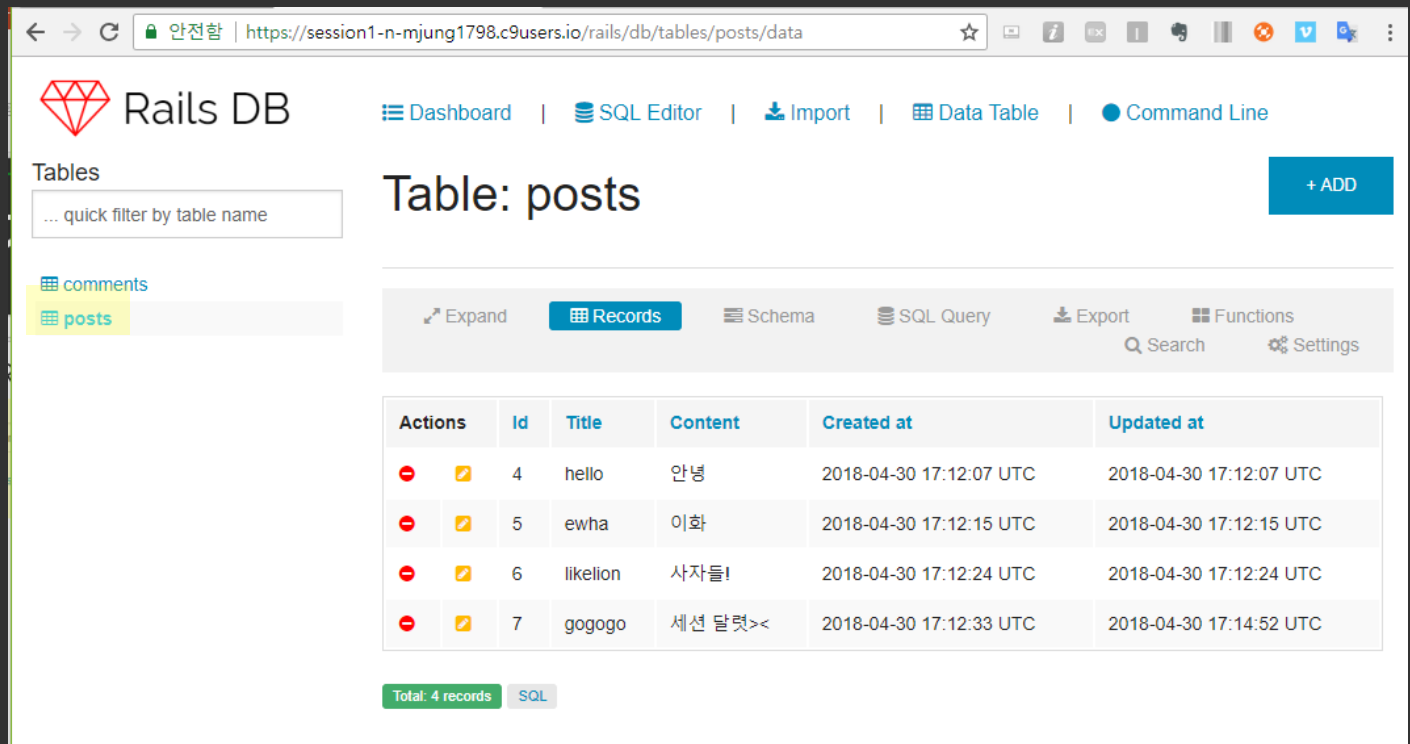
- use command line tools `railsdb` and `runsql` to run app and sql for your apps.
- view content and schema of DB tables (including indexes)
- create, edit content
- sort content of DB tables
- filters support
- execute SQL queries
- enjoy autocomplete for your SQL queries
- filter tables for easier navigation
- use history of your SQL queries
- see EXPLAIN results for your queries
- export data to CSV and Excel (see documentation)
- hide/show columns
- import SQL files
- expand/collapse sidebar
- experience added HTTP_BASIC auth for Rails DB
- ability to turn on/off using initializer
- include white/black lists
- explore data-tables inside your app
- ajax-ify Rails DB
- delete records, truncate tables
- __visualize SQL schema (future)__
- __CRUD for table records (future)__











LIKE LION

<생성된 모델 확인하기>

gem rails_db



The screenshot shows the Rails DB web interface. The browser address bar displays the URL: <https://session1-n-mjung1798.c9users.io/rails/db/tables/posts/data>. The interface includes a navigation bar with links for Dashboard, SQL Editor, Import, Data Table, and Command Line. On the left, a sidebar lists tables: 'comments' and 'posts' (highlighted). The main area is titled 'Table: posts' and features a '+ ADD' button. Below the title, there are tabs for 'Expand', 'Records' (selected), 'Schema', 'SQL Query', 'Export', 'Functions', 'Search', and 'Settings'. The 'Records' tab displays a table with 4 records. The table has columns: Actions, Id, Title, Content, Created at, and Updated at. The data rows are as follows:

Actions	Id	Title	Content	Created at	Updated at
 	4	hello	안녕	2018-04-30 17:12:07 UTC	2018-04-30 17:12:07 UTC
 	5	ewha	이화	2018-04-30 17:12:15 UTC	2018-04-30 17:12:15 UTC
 	6	likellion	사자들!	2018-04-30 17:12:24 UTC	2018-04-30 17:12:24 UTC
 	7	gogogo	세션 달렸><	2018-04-30 17:12:33 UTC	2018-04-30 17:14:52 UTC

At the bottom of the table, it says 'Total: 4 records' and 'SQL'.

Post모델에 저장한 모든 내용이 Table형식으로 정리!



LIKE LION

1) Comment 모델 만들기

```
rails g model comment content:string post_id: integer
```

- **Comment**라는 이름의 모델을 만들어낸다(generate)
- Type이 string인 **content** column
- Type이 integer인 **post_id** column



LIKE LION

1) Comment 모델 만들기

rails g model **comment** content:string post_id: integer

```
class CreateComments < ActiveRecord::Migration
  def change
    create_table :comments do |t|
      t.string :content
      t.integer :post_id

      t.timestamps null: false
    end
  end
end
```



LIKE LION

2) 모델 사이 관계 만들기

1개의 **게시글**에 n개의 댓글이 포함된다.

Post 모델에 명령어! <app/models/post.rb>

has_many :comments

```
post.rb
1 class Post < ActiveRecord::Base
2   has_many :comments
3 end
4
```



LIKE LION

2) 모델 사이 관계 만들기

n개의 댓글은 1개의 게시물에 속한다.

Comment 모델에 명령어! <app/models/comment.rb>

belongs_to :post

```
comment.rb
1 class Comment < ActiveRecord::Base
2   belongs_to :post
3 end
4
```



LIKE LION

2) 모델 사이 관계 만들기

<model/comment.rb>

belongs_to :post 여러 개의 comments들은 post **하나**에 속한다. (단수형)

<model/post.rb>

has_many :comments 하나의 post는 여러 개의 comments를 갖는다. (복수형)
여러 개의 comments를 소유하기 때문



LIKE LION

2) 모델 사이 관계 만들기

<model/comment.rb>

belongs_to :post 여러 개의 comments들은 post **하나**에 속한다. (단수형)

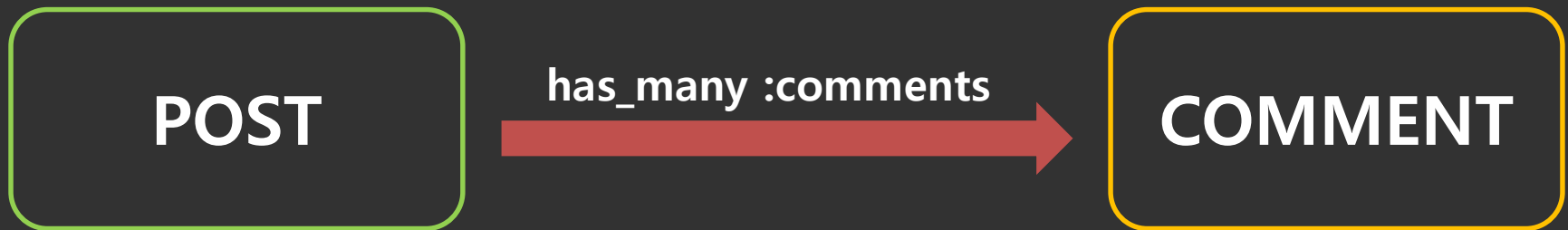
<model/post.rb>

has_many :comments 하나의 post는 여러 개의 comments를 갖는다. (복수형)
여러 개의 comments를 소유하기 때문



LIKE LION

2) 모델 사이 관계 만들기

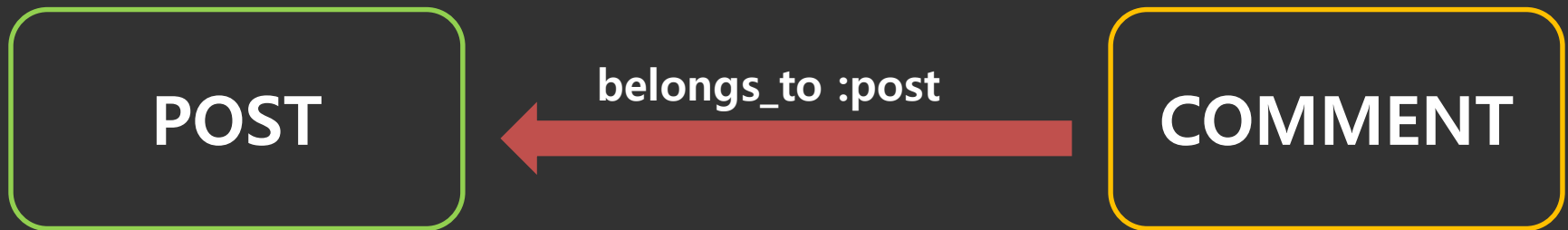


POST모델에서 COMMENT모델에 접근 할 수 있다.



LIKE LION

2) 모델 사이 관계 만들기

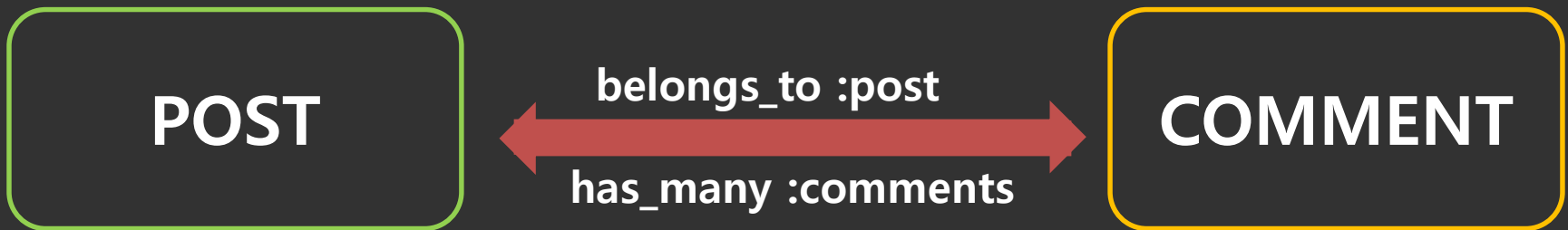


COMMENT모델에서 POST모델에 접근 할 수 있다.



LIKE LION

2) 모델 사이 관계 만들기



서로가 서로를 접근 할 수 있다.

어플리케이션 확장중에는 어떤 일이 있을지 몰라

두가지 모두 사용할 것을 추천!



LIKE LION

3) Controller & View 만들기

CRUD 기억하기!

Post모델을 해당(사용)하는 **MVC**모두 생성했었다!

VC생성

Rails g **controller home** new create index edit update destroy

Rails g **model post** title:text content:string

M생성



LIKE LION

3) Controller & View 만들기

rails g controller **comments** create destroy

같은 맥락으로 home controller에서 post 모델 관련 처리를 한 것처럼
Comments controller에서 comment모델 관련 처리를 하자



LIKE LION

3) Controller & View 만들기

rails g controller **comments** create destroy

- ▼ controllers
 - ▶ concerns
 - application_controller.rb
 - comments_controller.rb**
 - home_controller.rb

```
class CommentsController < ApplicationController
  def create
  end

  def destroy
  end
end
```

- ▼ views
 - ▼ comments**
 - create.html.erb
 - destroy.html.erb
 - ▼ home
 - create.html.erb
 - destroy.html.erb
 - edit.html.erb
 - index.html.erb
 - new.html.erb
 - ▶ layouts



LIKE LION

4) 댓글 입력 <index.html.erb>

```
<h1>여기는 게시물 목록이 나올 페이지 입니다!</h1>
<a href = "/home/new">새로운 글 쓰기</a>

<br><br>
<%@posts.each do |x|%>
  제목 : <%=x.title%><br>
  내용 : <%=x.content%><br>
  <a href = '/home/destroy/<%=x.id%>' style = 'text-decoration:none;'>[삭제]</a><br>
  <a href = '/home/edit/<%=x.id%>' style = 'text-decoration:none;'>[수정]</a><br>
  <hr>
  <form action = '/posts/<%=x.id%>/comments/create' method = "POST">
    <textarea name = "content"></textarea>
    <input type = "submit" value = "댓글쓰기"><br>
  </form>
<%end%>
```

댓글 입력부분 추가!

Action확인하기! => 해당 uri **route** 해야겠지?!



LIKE LION

4) 댓글 입력 <index.html.erb>

```
<form action='/posts/<%=x.id%>/comments/create' method ="POST">
```

메소드는 POST

'/posts/:post_id/comments/create' uri가 입력되면?



LIKE LION

4) 댓글 입력 <index.html.erb>

```
<form action='/posts/<%=x.id%>/comments/create' method ="POST">
```

Comment모델에 저장하는 작업을 해야하니까!

Comment들 관련한 작업을 수행하려고 만든

comments controller에서 액션을 수행해야한다!

아까 rails g controller **comments** create destroy했었다!



LIKE LION

5) 댓글 입력 route 설정 <routes.rb>

```
post '/posts/:post_id/comments/create' => 'comments#create'
```

'/posts/:post_id/comments/create' uri이 입력되면

comments controller의 **create 액션으로** 가줘!



LIKE LION

6) 댓글 입력 action 설정 <comments_controller.rb>

comments controller의 create 액션에서는?

- Comment **모델에 저장**하는 작업!!

```
class CommentsController < ApplicationController
  def create
    @comment = Comment.new
    @comment.content = params[:content]
    @comment.post_id = params[:post_id]
    @comment.save

    redirect_to "/home/index"
  end
end
```



LIKE LION

6) 댓글 입력 action 설정 <comments_controller.rb>

```
class CommentsController < ApplicationController
  def create
    @comment = Comment.new
    @comment.content = params[:content]
    @comment.post_id = params[:post_id]
    @comment.save

    redirect_to "/home/index"
  end
end
```

```
@comment = Comment.new
```

View에서도 쓸 수 있는 변수인 @comment 변수!
여기에 Comment모델의 한 줄의 row를 생성해줘!

Comment Model

id	content	post_id	Created_at
1			2018.05.03 19:00



LIKE LION

6) 댓글 입력 action 설정 <comments_controller.rb>

```
class CommentsController < ApplicationController
  def create
    @comment = Comment.new
    @comment.content = params[:content]
    @comment.post_id = params[:post_id]
    @comment.save

    redirect_to "/home/index"
  end
end
```

```
@comment.content = params[:content]
@comment.post_id = params[:post_id]
```

@comment 변수 (생성한 row)의
content column에 params[:content]를 넣을래
post_id column에 params[:post_id]를 넣을래

Comment Model

id	content	post_id	Created_at
1	params[:content]	params[:post_id]	2018.05.03 19:00

6) 댓글 입력 action 설정 <comments_controller.rb>

```
class CommentsController < ApplicationController
  def create
    @comment = Comment.new
    @comment.content = params[:content]
    @comment.post_id = params[:post_id]
    @comment.save

    redirect_to "/home/index"
  end
end
```

```
@comment.content = params[:content]
@comment.post_id = params[:post_id]
```

```
<textarea name = "content"></textarea>
```

```
post '/posts/:post_id/comments/create' => 'comments#create'
```

Comment Model

id	content	post_id	Created_at
1	params[:content]	params[:post_id]	2018.05.03 19:00



LIKE LION

6) 댓글 입력 action 설정 <comments_controller.rb>

```
class CommentsController < ApplicationController
  def create
    @comment = Comment.new
    @comment.content = params[:content]
    @comment.post_id = params[:post_id]
    @comment.save

    redirect_to "/home/index"
  end
end
```

```
@comment.save

redirect_to "/home/index"
```

작성한 이것들을 save해줘!

그 후 원래는 comments폴더의 create.html.erb로 넘어가지만
redirect_to를 이용해서 /home/index로 가줘

Comment Model

id	content	post_id	Created_at
1	params[:content]	params[:post_id]	2018.05.03 19:00



LIKE LION

7) 댓글 보기 view설정 <home/index.html.erb>

```
<h1>여기는 게시물 목록이 나올 페이지 입니다!</h1>
<a href = "/home/new" >새로운 글 쓰기</a>

<br><br>
<%@posts.each do |x|%>
  제목 : <%=x.title%><br>
  내용 : <%=x.content%><br>
  <a href = '/home/destroy/<%=x.id%>' style = 'text-decoration:none;'>[삭제]</a><br>
  <a href = '/home/edit/<%=x.id%>' style = 'text-decoration:none;'>[수정]</a><br>
  <hr>
  <form action = '/posts/<%=x.id%>/comments/create' method = "POST">
    <textarea name = "content"></textarea>
    <input type = "submit" value = "댓글쓰기"><br>
  </form>

  <%=x.comments.each do |comment|%>
    <%=comment.content%><br>
  <%end%>
  <hr>
<%end%>
```



LIKE LION

7) 댓글 보기 view설정 <home/index.html.erb>

```
<%@posts.each do |x|%>
```

```
<%x.comments.each do |comment|%>
```

```
<%=comment.content%><br>
```

```
<%end%>
```

```
class Post < ActiveRecord::Base
  has_many :comments
end
```



7) 댓글 보기 view설정 <home/index.html.erb>

```
<%@posts.each do |x|%>
```

```
<%x.comments.each do |comment|%>
```

```
<%=comment.content%><br>
```

```
<%end%>
```

```
class Post < ActiveRecord::Base
  has_many :comments
end
```

Post모델에서 Comment모델을 접근한다

```
<%x.comments
```

id	content
----	---------

1	마지막 시험까지 멋지게 조졌다ㅎㅎ
---	--------------------

id	content	post_id
----	---------	---------

5	ㅋㅋ 나두 어쨌든 끝났다 ~~	1
---	------------------	---

6	22맞아 어쨌거나 끝남 ~~	1
---	-----------------	---

7	물화 슈밤 ~~	1
---	----------	---



LIKE LION

7) 댓글 보기 view설정 <home/index.html.erb>

```
<%@posts.each do |x|%>
```

```
<%x.comments.each do |comment|%>
```

```
<%=comment.content%><br>
```

```
<%end%>
```

```
class Post < ActiveRecord::Base
  has_many :comments
end
```

Post모델에서 Comment모델을 접근한다

```
<%x.comments
```

id	content
----	---------

1	마지막 시험까지 멋지게 조졌다ㅎㅎ
---	--------------------

id	content	post_id
----	---------	---------

5	ㅋㅋ 나두 어쨌든 끝났다 ~~	1
---	------------------	---

6	22맞아 어쨌거나 끝남 ~~	1
---	-----------------	---

7	물화 슈밤 ~~	1
---	----------	---

해당 post_id를 가진 comment들을
1개씩 뽑는다!

```
.each do |comment|%>
```



LIKE LION

<확인해볼까? RUN!>

여기는 게시물 목록이 나올 페이지 입니다!

[새로운 글 쓰기](#)

제목 : hello

내용 : 안녕

[\[삭제\]](#)

[\[수정\]](#)

안녕!?

댓글쓰기

안녕!

안녕안녕!!

제목 : ewha

내용 : 이화

[\[삭제\]](#)

[\[수정\]](#)

댓글쓰기

제목 : likelion

내용 : 사자들!

[\[삭제\]](#)

[\[수정\]](#)

댓글쓰기

여기는 게시물 목록이 나올 페이지 입니다!

[새로운 글 쓰기](#)

제목 : hello

내용 : 안녕

[\[삭제\]](#)

[\[수정\]](#)

댓글쓰기

안녕!

안녕안녕!!

안녕!?

제목 : ewha

내용 : 이화

[\[삭제\]](#)

[\[수정\]](#)

댓글쓰기

제목 : likelion

내용 : 사자들!

[\[삭제\]](#)

[\[수정\]](#)

댓글쓰기



LIKE LION

<확인해볼까? RUN!>

Table: posts

[Expand](#)[Records](#)[Schema](#)[SQL Query](#)

Actions	Id	Title	Content	Created at
 	4	hello	안녕	2018-04-30 17:12:07 UTC
 	5	ewha	이화	2018-04-30 17:12:15 UTC
 	6	likelion	사자들!	2018-04-30 17:12:24 UTC
 	7	gogogo	세션 달렸><	2018-04-30 17:12:33 UTC

Total: 4 records

[SQL](#)

Table: comments

[Expand](#)[Records](#)[Schema](#)[SQL Query](#)

Actions	Id	Content	Post	Created at
 	1	안녕!	4	2018-05-01 05:07:42 UTC
 	2	안녕안녕!!	4	2018-05-01 05:07:48 UTC
 	3	안녕!?	4	2018-05-01 05:38:58 UTC

Total: 3 records

[SQL](#)

LIKE LION

8) 댓글 지우기 view설정 <home/index.html.erb>

CRUD Destroy와 같은 방법!!

```
<h1>여기는 게시물 목록이 나올 페이지 입니다!</h1>
<a href = "/home/new">새로운 글 쓰기</a>

<br><br>
<%@posts.each do |x|%>
  제목 : <%=x.title%><br>
  내용 : <%=x.content%><br>
  <a href = '/home/destroy/<%=x.id%>' style = 'text-decoration:none;'>[삭제]</a><br>
  <a href = '/home/edit/<%=x.id%>' style = 'text-decoration:none;'>[수정]</a><br>
  <hr>
  <form action = '/posts/<%=x.id%>/comments/create' method = "POST">
    <textarea name = "content"></textarea>
    <input type = "submit" value = "댓글 쓰기"><br>
  </form>

  <%x.comments.each do |comment|%>
    <%=comment.content%> <a href = "/posts/<%=x.id%>/comments/<%=comment.id%>">[삭제]</a>
  <br>
  <%end%>
  <hr>
<%end%>
```



LIKE LION

9) 댓글 지우기 route설정 <routes.rb>

```
get '/posts/:post_id/comments/:comments_id' => 'comments#destroy'
```

'/posts/:post_id/comments/:comments_id' uri가 입력되면

comments controller의 destroy 액션으로 가줘!



LIKE LION

10) 댓글 지우기 controller설정 <comments_controller.rb>

CRUD Destroy와 같은 방법!!

```
class CommentsController < ApplicationController
  def create
    @comment = Comment.new
    @comment.content = params[:content]
    @comment.post_id = params[:post_id]
    @comment.save

    redirect_to "/home/index"
  end

  def destroy
    destroy_comment = Comment.find(params[:comments_id])
    destroy_comment.destroy

    redirect_to "/home/index"
  end
end
```



LIKE LION

10) 댓글 지우기 controller설정 <comments_controller.rb>

CRUD Destroy와 같은 방법!!

```
def destroy
  destroy_comment = Comment.find(params[:comments_id])
  destroy_comment.destroy

  redirect_to "/home/index"
end
```

```
destroy_comment = Comment.find(params[:comments_id])
```

받은 comments_id가 있는 row를 Comment 모델에서 찾아줘!
해당하는 row를 destroy_comment 변수에 저장!



LIKE LION

10) 댓글 지우기 controller설정 <comments_controller.rb>

CRUD Destroy와 같은 방법!!

```
destroy_comment = Comment.find(params[:comments_id])
```

```
<a href="/posts/<%=x.id%>/comments/<%=comment.id%>">[삭제]</a>
```

id	content	post_id	Created_at
1	내말이 진짜 개박쳐..	1	2018.04.29. 23:33
2	کم공인갓	1	2018.04.30. 00:48
3	벗도 کم공인가	1	2018.04.30. 00:50
4	과제 팀플 너무 많아..	1	2018.04.30. 09:17
5	큁큁 나두 어쨌든 끝났다 ~~	2	2018.04.30. 10:36
6	22맞아 어쨌거나 끝남 ~~	2	2018.04.30. 10:40
7	물화 슈밤 ~~	2	2018.04.30. 13:25
8	시험기간->과제기간->시험기간->방학	1	2018.04.30. 21:23



LIKE LION

10) 댓글 지우기 controller 설정 <comments_controller.rb>

CRUD Destroy와 같은 방법!!

```
def destroy
  destroy_comment = Comment.find(params[:comments_id])
  destroy_comment.destroy

  redirect_to "/home/index"
end
```

```
destroy_comment = Comment.find(params[:comments_id])
```

받은 comments_id가 있는 row를 Comment 모델에서 찾아줘!
해당하는 row를 destroy_comment 변수에 저장!

```
destroy_comment.destroy
```

해당 row를 지우기!!



LIKE LION

10) 댓글 지우기 controller설정 <comments_controller.rb>

CRUD Destroy와 같은 방법!!

```
def destroy
  destroy_comment = Comment.find(params[:comments_id])
  destroy_comment.destroy

  redirect_to "/home/index"
end
```

```
destroy_comment = Comment.find(params[:comments_id])
```

받은 comments_id가 있는 row를 Comment 모델에서 찾아줘!
해당하는 row를 destroy_comment 변수에 저장!

```
destroy_comment.destroy
```

해당 row를 지우기!!

```
redirect_to "/home/index"
```

원래는 comments폴더의 destroy.html.erb로 넘어가지만
redirect_to를 이용해서 /home/index로 가줘



LIKE LION

Part 4 . 정 리



LIKE LION

Routes의 역할!

```
get '/posts/:post_id/comments/:comments_id'=>'comments#destroy'
```

Uri를 내마음대로 지정해서 원하는 action을 할 수 있다

```
<%x.comments.each do |comment|%>  
<%=comment.content%> <a href="/posts/<%=x.id%>/comments/<%=comment.id%>">[삭제]</a>
```

view

```
get '/posts/:post_id/comments/:comments_id'=>'comments#destroy'
```

routes

```
destroy_comment = Comment.find(params[:comments_id])
```

controller

:comment_id :post_id 지정!!



LIKE LION


```
<%@posts.each do |x|%>
```

Post모델에서 Comment모델을 **접근**한다

```
<%x.comments.each do |comment|%>
```

```
<%x.comments
```

```
<%=comment.content%><br>
```

```
<%end%>
```

Comment 모델에서 post_id column을 **저장**했기에 가능했다!

Post에서 comment 접근할 때 자동으로 post_id를 찾는다!

따라서 post_id는 꼭 이 이름의 column으로 지정

Table: posts















Expand Records Schema SQL Query				
Actions	Id	Title	Content	Created at
 	4	hello	안녕	2018-04-30 17:12:07 UTC
 	5	ewha	이화	2018-04-30 17:12:15 UTC
 	6	likelion	사자들!	2018-04-30 17:12:24 UTC
 	7	gogogo	세션 달렸><	2018-04-30 17:12:33 UTC
Total: 4 records SQL				

Table: comments

Expand Records Schema SQL Query				
Actions	Id	Content	Post	Created at
 	1	안녕!	4	2018-05-01 05:07:42 UTC
 	2	안녕안녕!!	4	2018-05-01 05:07:48 UTC
 	3	안녕!?	4	2018-05-01 05:38:58 UTC
Total: 3 records SQL				



LIKE LION

1) Comment 모델 만들기

2) 모델 사이 관계 만들기

3) Controller & View 만들기

- C { 4) 댓글 입력 <index.html.erb>
- 5) 댓글 입력 route 설정<routes.rb>
- 6) 댓글 입력 action설정<comments_controller.rb>
- R { 7) 댓글 보기 view설정<home/index.html.erb>
- 8) 댓글 지우기 view설정<home/index.html.erb>
- D { 9) 댓글 지우기 route설정<routes.rb>
- 10) 댓글 지우기 controller설정<comments_controller.rb>



LIKE LION

댓글에서도 CRUD가 적용!
다른건 모델에서의 1:N관계 적용뿐!



LIKE LION

그럼 댓글에서의 U는??



LIKE LION

6기 스스로 해보기 >< 파이팅!!



LIKE LION