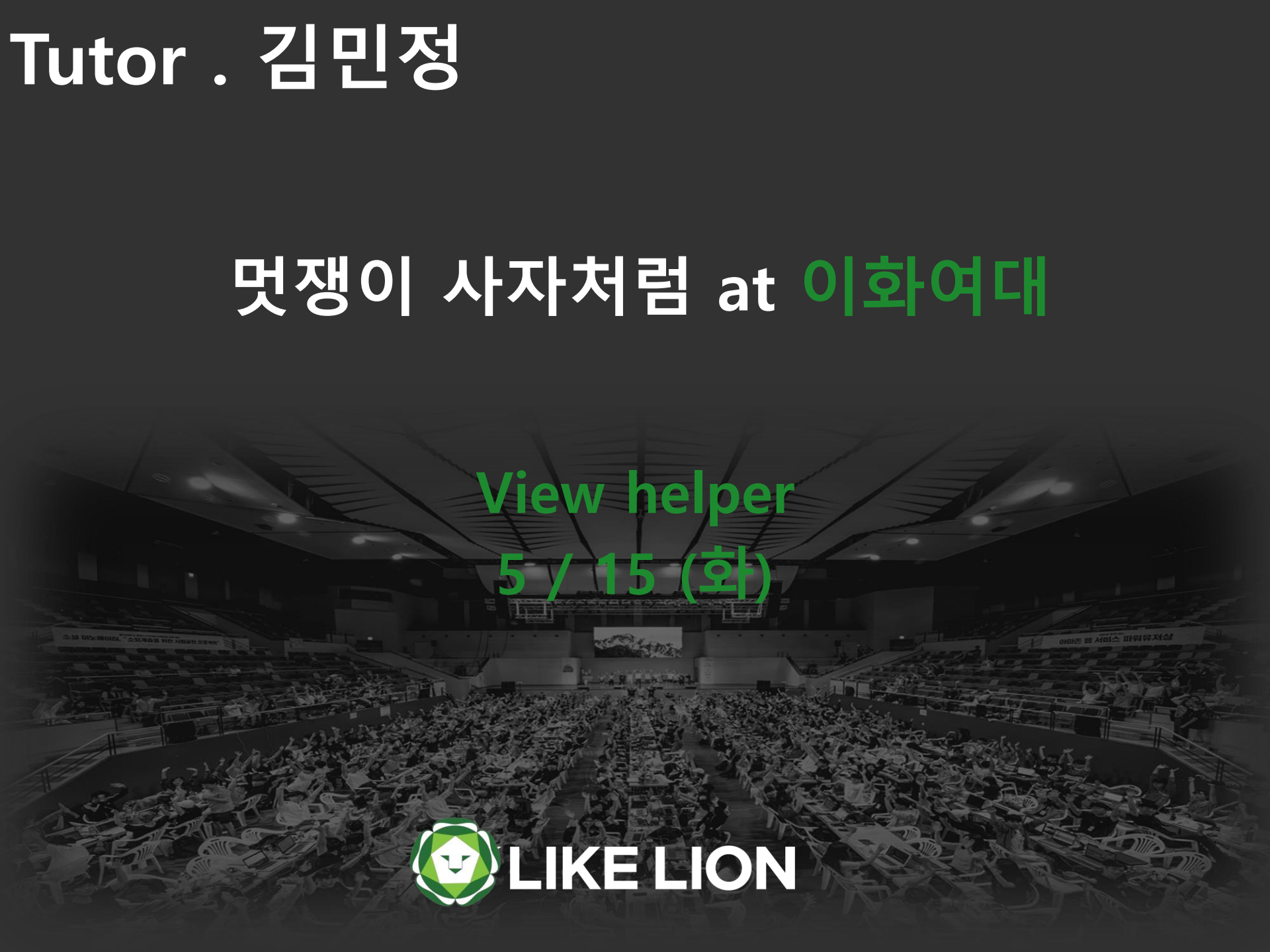
Tutor . 김민정

멋쟁이 사자처럼 at 이화여대



Part 1 . View helper

그게 뭐에요?

LIKE LION

<%= %> <% %>

.html.erb 파일에서 ruby문법을 쓰기 위해 썼던 저 괄호!!

View에서 ruby문법 적용하기!

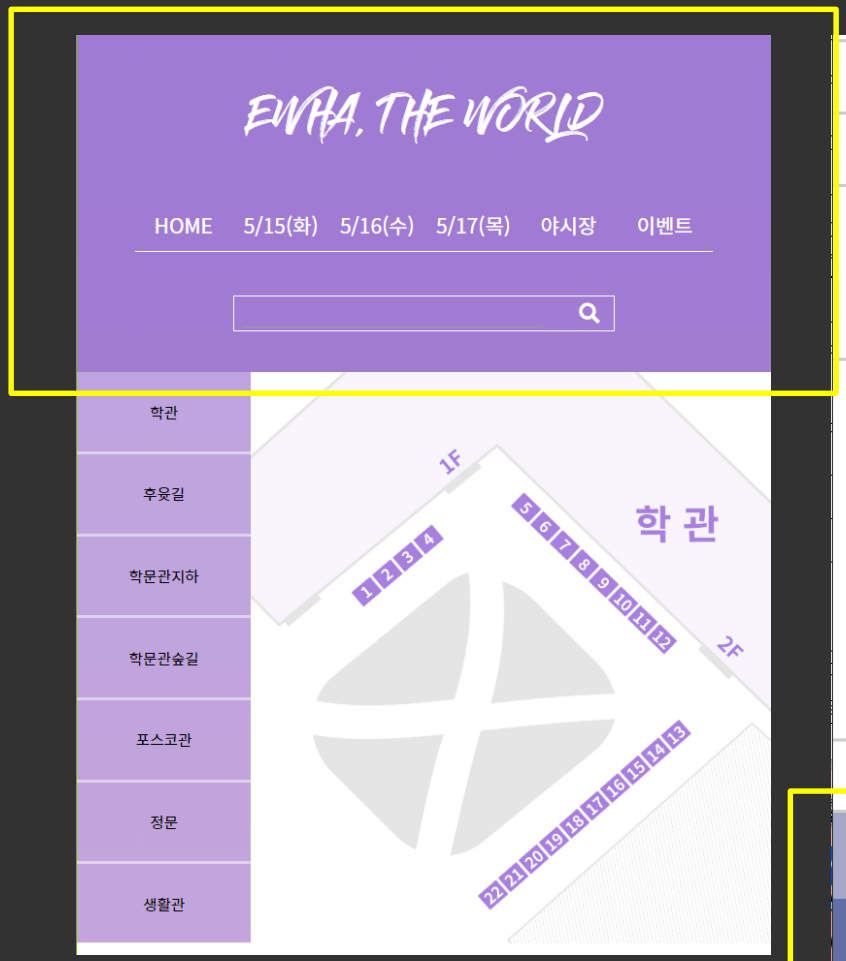
LIKE LION

이거 왜 쓰는거지

LIKE LION

축제 사이트 개발

LIKE LION


EWHA, THE WORLD
HOME
5/15(화)
5/16(수)
5/17(목)
야시장
이벤트
학관
후윳길
학문관지하
학문관숲길
포스코관
정문
생활관
1F
2F
학 관
1 2 3 4
5 6 7 8 9 10 11 12
22 21 20 19 18 17 16 15 14 13


19 말랑말랑한 뇌
20 JDM
21 국어국문학과 학생회
떡꼬치 : 1000
22 영어영문학과 학생회
핫도그 : 3000
청포도에이드 : 2000
세트 : 4500
Enter sold out password
Click Me
Copyright © Ewha-Likelion 6th, 2018

LIKE LION

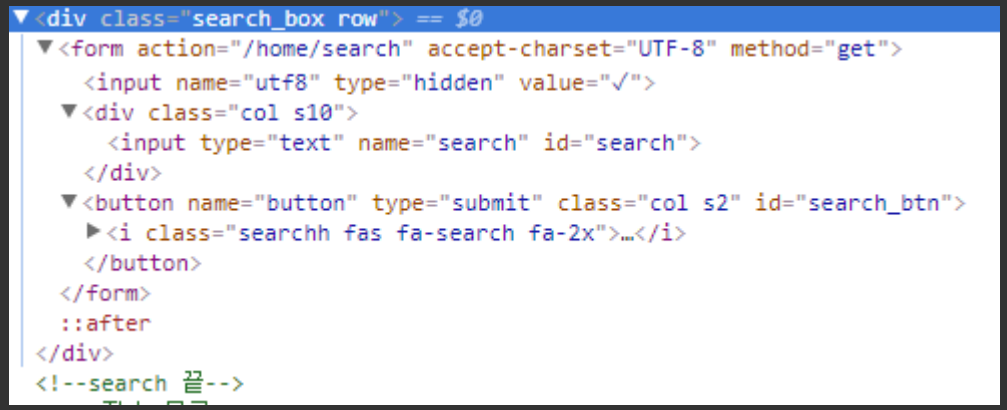
[application.html.erb]

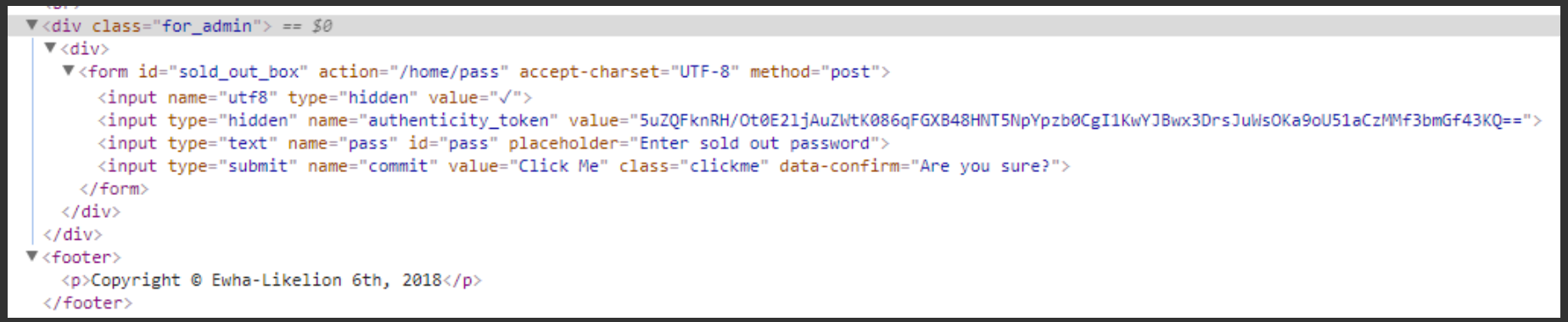
```
67    <!--search-->
68    <div class="search_box row">
69      <%= form_tag "/home/search" , method: "get" do %>
70          <div class ="col s10"><%= text_field_tag "search"%></div>
71          <%= button_tag(type: "submit", class: "col s2", id: "search_btn") do %>
72                <i class="searchh fas fa-search fa-2x"></i>
73          <% end %>
74      <% end %>
75      </div>
76      <!--search 끝-->
77
78
79
80  <%= yield %>
81
82  <!--password-->
83
84  <br>
85      <div class="for_admin">
86          <div>
87              <%= form_tag "/home/pass", :id => 'sold_out_box' do %>
88                  <%= text_field_tag "pass", nil, :placeholder => 'Enter sold out password' %>
89                  <%= submit_tag "Click Me", :class => 'clickme', :data => { :confirm => "Are you su
90              <% end %>
91          </div>
92      </div>
93
94  <!--password 끝-->
95
96      <footer>
97          <p>Copyright &copy; Ewha-Likelion 6th, 2018</p>
98      </footer>
99
```

[application.html.erb]

```
67  <!--search-->
68  <div class="search_box row">
69    <%= form_tag "/home/search" , method: "get" do %>
70        <div class ="col s10"><%= text_field_tag "search"%></div>
71        <%= button_tag(type: "submit", class: "col s2", id: "search_btn") do %>
72              <i class="searchh fas fa-search fa-2x"></i>
73        <% end %>
74    <% end %>
75    </div>
76    <!--search 끝-->
77
78
79
80  <%= yield %>
81
82  <!--password-->
83
84  <br>
85      <div class="for_admin">
86          <div>
87              <%= form_tag "/home/pass", :id => 'sold_out_box' do %>
88                  <%= text_field_tag "pass", nil, :placeholder => 'Enter sold out password' %>
89                  <%= submit_tag "Click Me", :class => 'clickme', :data => { :confirm => "Are you su
90              <% end %>
91          </div>
92      </div>
93
94  <!--password 끝-->
95
96      <footer>
97          <p>Copyright &copy; Ewha-Likelion 6th, 2018</p>
98      </footer>
99
```

LIKE LION





루비 문법으로 html 태그를 만들 수 있다!

LIKE LION

1. 사이트의 전체적인 디자인을 한꺼번에 적용할 수 있다.

2. 개별 템플릿에 페이지 자체의 내용만 작성할 수 있다.

3. 사이트 구조에 일관성을 부여할 수 있다.

즉, html로 여러 번 쓸 것을 helper를 이용해서 한 번에 쓴다!

Html : 정적인 text Helper : 동적인 ruby 코드

여기서 잠깐

Part 2 . tag

LIKE LION

[application.html.erb]

```
67    <!--search-->
68    <div class="search_box row">
69      <%= form_tag "/home/search" , method: "get" do %>
70          <div class ="col s10"><%= text_field_tag "search"%></div>
71          <%= button_tag(type: "submit", class: "col s2", id: "search_btn") do %>
72                <i class="searchh fas fa-search fa-2x"></i>
73          <% end %>
74      <% end %>
75      </div>
76      <!--search 끝-->
77
78
79
80  <%= yield %>
81
82  <!--password-->
83
84  <br>
85      <div class="for_admin">
86          <div>
87              <%= form_tag "/home/pass", :id => 'sold_out_box' do %>
88                  <%= text_field_tag "pass", nil, :placeholder => 'Enter sold out password' %>
89                  <%= submit_tag "Click Me", :class => 'clickme', :data => { :confirm => "Are you su
90              <% end %>
91          </div>
92      </div>
93
94  <!--password 끝-->
95
96      <footer>
97          <p>Copyright &copy; Ewha-Likelion 6th, 2018</p>
98      </footer>
99
```

[application.html.erb]

```
67   <!--search-->
68   <div class="search_box row">
69     <%= form_tag "/home/search" , method: "get" do %>
70         <div class ="col s10"><%= text_field_tag "search"%></div>
71         <%= button_tag(type: "submit", class: "col s2", id: "search_btn") do %>
72               <i class="searchh fas fa-search fa-2x"></i>
73         <% end %>
74     <% end %>
75     </div>
76     <!--search 끝-->
```

```
87              <%= form_tag "/home/pass", :id => 'sold_out_box' do %>
88                  <%= text_field_tag "pass", nil, :placeholder => 'Enter sold out password' %>
89                  <%= submit_tag "Click Me", :class => 'clickme', :data => { :confirm => "Are you su
90              <% end %>
```

엇 html의 태그랑 비슷한 이 view helper는 뭐지?

축제 사이트 : (URL)

F12 (개발자 도구)를 눌러보자!

LIKE LION

자주쓰는 view helper tag

<%= link_to %>

<%= Form_tag %>

<%= Image_tag %>

LIKE LION

C9 create! / git clone

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

Part 3 . Link_to

<a>와 같은 역할

<%= link_to %>

LIKE LION

<%= link_to '텍스트', 'URL'%>

LIKE LION

<%= link_to '텍스트', 'URL'%>

<a>텍스트

LIKE LION

텍스트

<%= link_to '텍스트', 'URL'%>

<a>텍스트

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>
```

```
get 'home/destroy/:post_id' => 'home#destroy'
```
[Routes.rb]

1. url을 직접 입력

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>
```

```
get 'home/destroy/:post_id' => 'home#destroy'
```
[Routes.rb]

1. url을 직접 입력

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>
```

URL 텍스트

```
get 'home/destroy/:post_id' => 'home#destroy'
```

[Routes.rb]

```
<%= link_to '삭제', "home/destroy/#{x.id}"%>
```

텍스트 URL

#{ }

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>
```

```
<%= link_to '삭제', "home/destroy/#{x.id}"%>
```

#{ }

```
"home/destroy/#{x.id}"
```

Helper를 써야하는 상태에서

“문자열”안에 변수, 값을 넣으려면 #{} 로 감싸준다

주의 : ‘’(작은 따옴표)는 안되고 “”(큰 따옴표)안에서만 가능

#{ }

```
"home/destroy/#{x.id}"
```

x.Id 값에 따라서

Home/destroy/1

Home/destroy/2

등등으로 이동

“home/destroy/x.id”

문자 그대로 ~/x.id URL로 이동

#{ }

```
<%= link_to '삭제', "home/destroy/#{x.id}"%>

<%= link_to '삭제', "https://view-helper-mjung1798.c9users.io/home/destroy/#{x.id}"%>
```

이 앞 부분은 항상 바뀔 수 있는 부분이므로 root URL의 뒷부분만 작성!

2. 컨트롤러와 액션을 명시

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>
```

```
<%= link_to '삭제', "/home/destroy/#{x.id}"%>
```

```
get 'home/destroy/:post_id' => 'home#destroy'
```

[Routes.rb]

home 컨트롤러의 destroy 액션이 처리한다!

2. 컨트롤러와 액션을 명시

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>
```

```
<%= link_to '삭제', "/home/destroy/#{x.id}"%>
```

```
get 'home/destroy/:post_id' => 'home#destroy'
```

[Routes.rb]

home 컨트롤러의 destroy 액션이 처리한다!

2. 컨트롤러와 액션을 명시

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>
```

```
<%= link_to '삭제', "/home/destroy/#{x.id}"%>
```

```
get 'home/destroy/:post_id' => 'home#destroy'
```
[Routes.rb]

home 컨트롤러의 destroy 액션이 처리한다!

```
<%= link_to '삭제', {controller: :home, action: :destroy, post_id: x.id}%>
```

LIKE LION

2. 컨트롤러와 액션을 명시

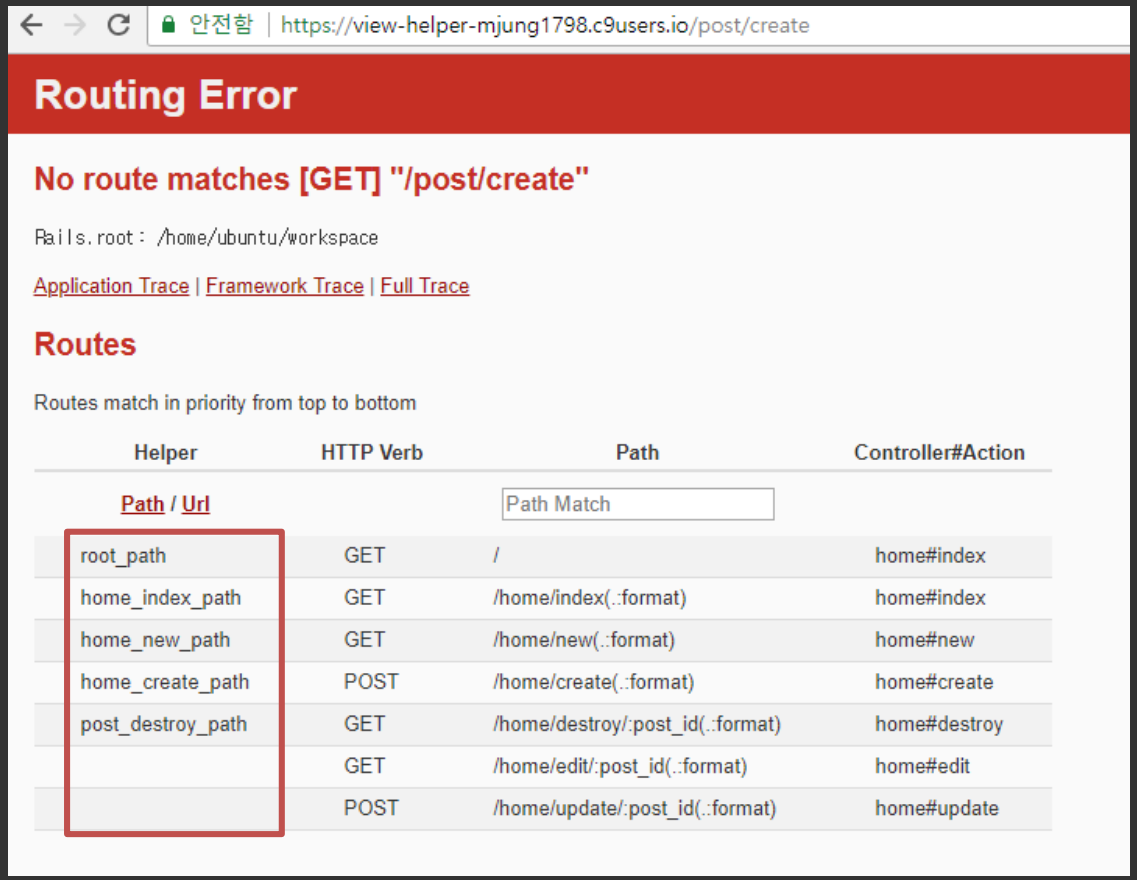
```
get 'home/destroy/:post_id' => 'home#destroy'
```
[Routes.rb]

post의 id값을 post_id로 지정한다!
이 post_id는 url로 해당 값을 넘겨줄 때 사용된다

```
<%= link_to '삭제', {controller: :home, action: :destroy, post_id: x.id}%>
```

주의! :home / :destroy지만 :x.id가 아니다!

3. URL에 이름을 붙여서 사용 ★★★



3. URL에 이름을 붙여서 사용 ★★★

```
<%= link_to '삭제', post_destroy_path(post_id: x.id)%>
```

```
get 'home/destroy/:post_id' => 'home#destroy'
```

[Routes.rb]

3. URL에 이름을 붙여서 사용 ★★★

```
<%= link_to '삭제', post_destroy_path(post_id: x.id)%>
```

```
get 'home/destroy/:post_id' => 'home#destroy', as: 'post_destroy'
```

[Routes.rb]

3. URL에 이름을 붙여서 사용 ★★★

```
<%= link_to '삭제', post_destroy_path(post_id: x.id)%>
```

```
get 'home/destroy/:post_id' => 'home#destroy', as: 'post_destroy'
```

[Routes.rb]

이 route에 post_destroy라는 **이름**을 준다

3. URL에 이름을 붙여서 사용 ★★★

```
<%= link_to '삭제', post_destroy_path(post_id: x.id)%>
```

Route에서 이름 붙인 것에 '_path'를 이어서 써준다

```
get 'home/destroy/:post_id' => 'home#destroy', as: 'post_destroy'
```

[Routes.rb]

3. URL에 이름을 붙여서 사용 ★★★

```
<%= link_to '삭제', post_destroy_path(post_id: x.id)%>
```

```
get 'home/destroy/:post_id' => 'home#destroy', as: 'post_destroy'
```

[Routes.rb]

Route에서 **post_id**라는 이름으로 **x.id**값을 보낸다고 했으니까!
(post_id: x.id)로 값을 보내준다

<link_to 정리>

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>

<%= link_to '삭제', "home/destroy/#{x.id}"%>

<%= link_to '삭제', {controller: 'home', action: 'destroy', post_id: x.id}%>

<%= link_to '삭제', post_destroy_path(post_id: x.id)%>
```

<link_to 정리>

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>

<%= link_to '삭제', "home/destroy/#{x.id}"%>

<%= link_to '삭제', {controller: 'home', action: 'destroy', post_id: x.id}%>

<%= link_to '삭제', post_destroy_path(post_id: x.id)%>
```

<%= link_to '텍스트', 'URL'%>

link_to 꾸미기 – 1. class 주기

```
<a href="URL" class="likelion">텍스트</a>
```

```
<%= link_to '텍스트', 'URL', class: "likelion"%>
```

link_to 꾸미기 – 1. class 주기

```
<a href ="/home/destroy/<%= x.id%>" class="likelion">삭제</a>

<%= link_to '삭제', "home/destroy/#{x.id}", class: "likelion"%>

<%= link_to '삭제', {controller: 'home', action: 'destroy', post_id: x.id}, class: "likelion"%>

<%= link_to '삭제', post_destroy_path(post_id: x.id), class: "likelion"%>
```

<%= link_to ‘텍스트’, ‘URL’, class: “likelion”%>

link_to 꾸미기 – 1. class 주기

```
<a href ="/home/destroy/<%= x.id%>" class="likelion">삭제</a>

<%= link_to '삭제', "home/destroy/#{x.id}", class: "likelion"%>

<%= link_to '삭제', {controller: 'home', action: 'destroy', post_id: x.id}, class: "likelion"%>

<%= link_to '삭제', post_destroy_path(post_id: x.id), class: "likelion"%>
```

<%= link_to ‘텍스트’, ‘URL’, class: “likelion”%>

link_to 꾸미기 – 2. 여러 태그넣기

<%= link_to %>에

텍스트 말고 다른 tag들 넣기

link_to 꾸미기 – 2. 여러 태그넣기

<a>

```
<a href ="/home/destroy/<%= x.id%>" class="likelion">
    <button>삭제</button>
</a>
```

<a>로 감싼다!

<%= link_to %>

```
<%= link_to '삭제', "home/destroy/#{x.id}", class: "likelion" do%>
    <button>삭제</button>
<%end%>
```

do end 로 감싼다!

LIKE LION

link_to 꾸미기 – 2. 여러 태그넣기

<%= link_to %> do end 로 감싼다!

```
<%= link_to "home/destroy/#{x.id}", class: "likelion" do%>
    <button>삭제</button>
<%end%>
```

link_to 꾸미기 – 2. 여러 태그넣기

<%= link_to %> do end 로 감싼다!

```
<%= link_to "home/destroy/#{x.id}", class: "likelion" do%>
    <button>삭제</button>
<%end%>
```

'URL' 'class'

```
<%= link_to "home/destroy/#{x.id}", class: "likelion" do%>
    <button>삭제</button>
<%end%>
```

link_to 꾸미기 – 2. 여러 태그넣기

<%= link_to %> do end 로 감싼다!

```
<%= link_to "home/destroy/#{x.id}", class: "likelion" do%>
    <button>삭제</button>
<%end%>
```

'URL' 'class'

```
<%= link_to "home/destroy/#{x.id}", class: "likelion" do%>
    <button>삭제</button>
<%end%>
```

<%= link_to ~~'텍스트'~~, 'URL', class:"likelion" do%>

<원하는 html tag!>

<%end%>

LIKE LION

Part 4 . Form_tag

<form>태그와 같은 역할

<%= form_tag %>

```
<form action ='/home/create' method ="post">
```

<form action="정보를 보낼 링크" method="방법">

LIKE LION

```
<form action ='/home/create' method ="post">
```

<form action="정보를 보낼 링크" method="방법">

```
<%= form_tag("/home/create", method: "post") do%>
```

<%= form_tag("URL", method: "post/get") do %>

<%end%>

LIKE LION

```
<form action ='/home/create' method ="post">
```

<form action="정보를 보낼 링크" method="방법">

```
<%= form_tag("/home/create", method: "post") do%>
```

방법

<%= form_tag("URL", method: "post/get") do %>

'정보를 보낼 주소'

Input을 감싸니까 do

<%end%>

LIKE LION

```
<form action ='/home/create' method ="post">
    <input type='text' name ='post_title'><br><br>
    <textarea name ='post_content'></textarea><br><br>
    <input type='submit' value='제출'>
</form>
```

```
<%= form_tag("/home/create", method: "post") do%>
    <input type='text' name ='post_title'><br><br>
    <textarea name ='post_content'></textarea><br><br>
    <input type='submit' value='제출'>
<%end%>
```

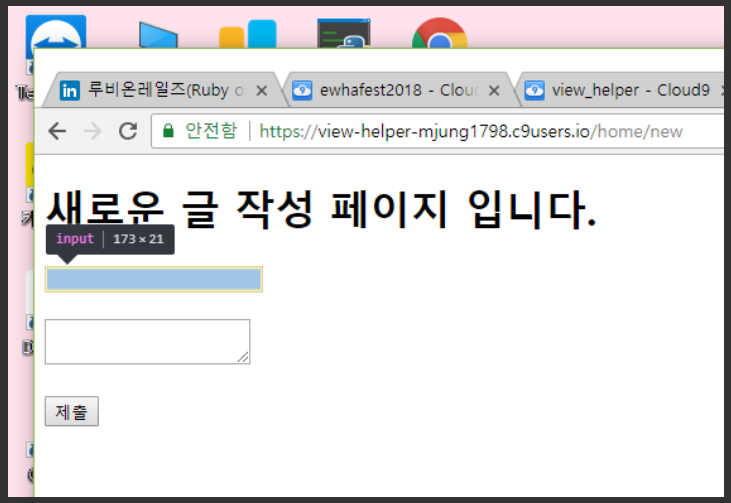
LIKE LION

왜 굳이?!

LIKE LION

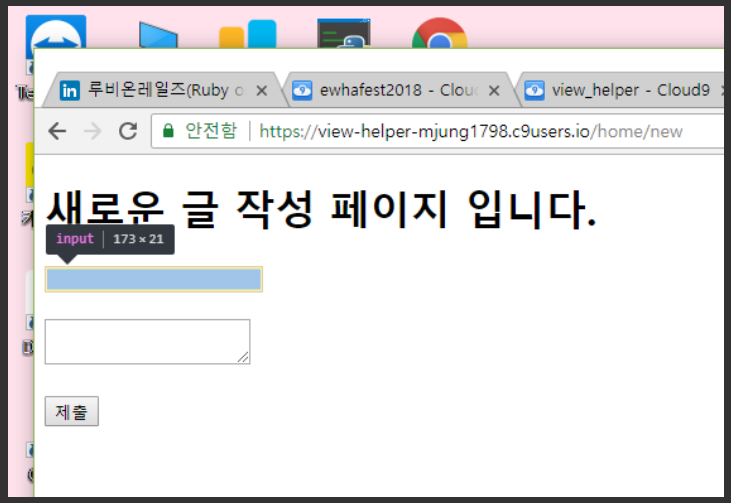
[f12 개발자도구]

```
▼<form action="/home/create" accept-charset="UTF-8" method="post"> == $0
  <input name="utf8" type="hidden" value="✓">
  <input type="hidden" name="authenticity_token" value="Ryazj6oOh6B5vB9v1zAkCDBO4XwyCQTnxiFEQ1Id2kh2Xf/NHRK1TM1I5d9iBRuuU/
  YJOBhsV2ViTrW4/tacnw==">
  <input type="text" name="post_title">
  <br>
  <br>
  <textarea name="post_content"></textarea>
  <br>
  <br>
  <input type="submit" value="제출">
</form>
```



[f12 개발자도구]

```
▼<form action="/home/create" accept-charset="UTF-8" method="post"> == $0
  <input name="utf8" type="hidden" value="✓">
  <input type="hidden" name="authenticity_token" value="Ryazj6oOh6B5vB9v1zAkCDBO4XwyCQTnxiFEQ1Id2kh2Xf/NHRK1TM1I5d9iBRuuU/
  YJOBhsV2ViTrW4/tacnw==">
  <input type="text" name="post_title">
  <br>
  <br>
  <textarea name="post_content"></textarea>
  <br>
  <br>
  <input type="submit" value="제출">
</form>
```

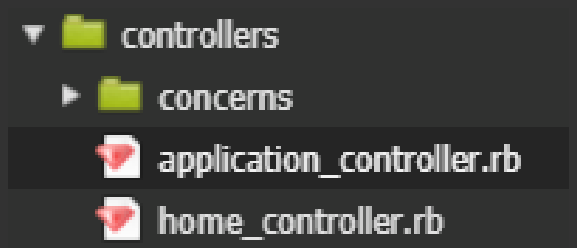


자동으로 2개의 input추가

```
<input type="hidden" name="authenticity_token"
value=
"HRmiA5OnEPPtowE1ndKX0Oc0W+1a03r+Sm++gKUUHEAsYu5BJL
iH11X+5Uo56h2hIyzrXC2KXzuAE96Cd9a1w==">
```

보안 문제!

```
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception
end
```



contollers/application_controller.rb

```ruby
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception
end
```

```html
<form action ='/home/create' method ="post">
    <input type='text' name ='post_title'><br><br>
    <textarea name ='post_content'></textarea><br><br>
    <input type='submit' value='제출'>
</form>
```

ActionController::InvalidAuthenticityToken in HomeController#create

ActionController::InvalidAuthenticityToken

Extracted source (around line #181):

```
179
180           def handle_unverified_request
181             raise ActionController::InvalidAuthenticityToken
182           end
183         end
184       end
```

Rails.root: /home/ubuntu/workspace

ActionController::InvalidAuthenticityToken in HomeController#create

ActionController::InvalidAuthenticityToken

Extracted source (around line #181):

```
179
180         def handle_unverified_request
181           raise ActionController::InvalidAuthenticityToken
182         end
183       end
184     end
```

Rails.root: /home/ubuntu/

```
<input type="hidden" name="authenticity_token"
value=
"HRmiA5OnEPPtowE1ndKX0Oc0W+1a03r+Sm++gKUUHEAsYu5BJL
iH11X+5Uo56h2hIyzrXC2KXzuAE96Cd9alw==">
```

```ruby
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception
end
```

Form 태그에서 날라온 값들 중에
authenticity_token이름으로 정확한 값이
날라왔는지 확인해준다.

html의 <form>태그는 토큰 값을 자동으로 만들어주지 못해서
일단 주석처리하고 작업했다!
<%=form_tag%>를 쓰면 토큰 input을 자동 생성해줘서 해결!

```ruby
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  # protect_from_forgery with: :exception
end
```

왜 토큰 값이 필요할까..?

```ruby
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  # protect_from_forgery with: :exception
end
```

왜 토큰 값이 필요할까..?

해커가 내 사이트에 마음대로

게시글을 쓰거나 글을 삭제할 수 있다.

route

```
get 'home/destroy/:post_id' => 'home#destroy', as: 'post_destroy'

get 'home/edit/:post_id'=>'home#edit'

post 'home/update/:post_id' => 'home#update'
```

내 사이트 뿐만 아니라 외부에서

왼쪽 URL대로 요청을 보내기만 한다면

해당 요청을 모두 처리해버림

```
<%= link_to '텍스트', 'URL'%>
```

```
<%= form_tag("URL", method: "post/get") do %>

<%end%>
```

```
<%= link_to '텍스트', 'URL'%>


<%= form_tag("URL", method: "post/get") do %>

<%end%>
```

<%= link_to ‘텍스트’, ‘URL’%>

엇 대체 할 수 있지 않을까? link_to의 3가지 방법!

<%= form_tag(“URL”, method: “post/get”) do %>

<%end%>

```
<a href ="/home/destroy/<%= x.id%>">삭제</a>

<%= link_to '삭제', "home/destroy/#{x.id}"%>

<%= link_to '삭제', {controller: 'home', action: 'destroy', post_id: x.id}%>

<%= link_to '삭제', post_destroy_path(post_id: x.id)%>
```

1. URL 직접 입력
2. 컨트롤러와 액션을 명시
3. URL에 이름을 붙여 사용

[실습!]

<form action =“/home/create” method= “post”>

1. veiw헬퍼로 만들기
2. Action부분을 link_to의 3가지 방법으로!

[solution!]

solution 1. URL 직접 입력

```
<%= form_tag("/home/create", method: "post") do%>
    <input type='text' name ='post_title'><br><br>
    <textarea name ='post_content'></textarea><br><br>
    <input type='submit' value='제출'>
<%end%>
```

solution 2. 컨트롤러와 액션을 명시

```
<%= form_tag({contoller:"home", action:"create"}, method: "post") do%>
    <input type='text' name ='post_title'><br><br>
    <textarea name ='post_content'></textarea><br><br>
    <input type='submit' value='제출'>
<%end%>
```

solution 3. URL에 이름을 붙여 사용

```
<%= form_tag(home_create_path, method: "post") do%>
    <input type='text' name ='post_title'><br><br>
    <textarea name ='post_content'></textarea><br><br>
    <input type='submit' value='제출'>
<%end%>
```

Part 5 . text_field_tag

<input>태그와 같은 역할

<%= text_field_tag %>

```
<input type='text' name ='post_title' placeholder="제목">
```

<input type="태그의 종류", name ="태그의 이름" >

```
<input type='text' name ='post_title' placeholder="제목">
```

<input type="태그의 종류", name ="태그의 이름" >

```
<%= text_field_tag "post_title", nil, placeholder: "제목"%>
```

<%= text_field_tag "태그의 이름", nil%>

```
<%= text_field_tag "post_title", nil, placeholder: "제목"%>
```

<%= text_field_tag “태그의 이름”, nil, placeholder: “제목”%>

태그의 이름 자체가 태그 종류

종류가 많으니 필요할때마다 찾기

바로 뒤에 붙는 “”가 name 값

```
<%= text_field_tag "post_title", nil, placeholder: "제목"%>
```

<%= text_field_tag “태그의 이름”, nil, placeholder: “제목”%>

두번째 값으로 오는건 input의 value=“” 속성!

즉, 처음부터 기본값이 들어가게 할 때 설정!

안할꺼면 nil, 할꺼면 “기본값” 이렇게 쓰기

```
<%= text_field_tag "post_title", nil, placeholder: "제목"%>
```

<%= text_field_tag “태그의 이름”, nil, placeholder: “제목”%>

이후에 class style등의 부가적인 속성!

이제 남은건?

<%=image_tag%>

<%=form_for%>

<%=text_area%>

<%=button_tag%>

<input>

해커톤 때 구글링으로 해결하기

끝!!!

-힘들어요... 자러갈꺼에요...