로그인 기능

with scaffold & devise gem

멋쟁이사자처럼 6기 운영진 | 김연진

scaffold

01 Scaffold

1. 그게 몰까?

Scaffolding is a technique supported by some model-view-controller frameworks, in which the programmer may write a specification that describes how the application database may be used. The compiler uses this specification to generate code that the application can use to create, read, update and delete database entries, effectively treating the template as a "scaffold" on which to build a more powerful application. Scaffolding is an evolution of database code generators from earlier development environments, such as Oracle's CASE Generator, and many other 4GL client-server software development products.
Scaffolding was popularized by the Ruby on Rails framework. It has been adapted to other software frameworks, including Django, Monorail (.Net), Symfony, CodeIgniter, Yii, CakePHP, Model-Glue, Grails, Catalyst, Seam Framework, Spring Roo, ASP.NET Dynamic Data and ASP.NET MVC Framework's Metadata Template Helpers.

http://blog.saltfactory.net/speedly-developing-web-service-via-scaffolding/
https://en.wikipedia.org/wiki/Scaffold_(programming)

멋쟁이사자처럼 @연세대 김연진

01 Scaffold

1. 그게 몰까?

Scaffolding is a technique supported by some model-view-controller frameworks, in which the programmer may write a specification that describes how the application database may be used. The compiler uses this specification to generate code that the application can use to create, read, update and delete database entries, effectively treating the template as a "scaffold" on which to build a more powerful application. Scaffolding is an evolution of database code generators from earlier development environments, such as Oracle's CASE Generator, and many other 4GL client-server software development products.
Scaffolding was popularized by the Ruby on Rails framework. It has been adapted to other software frameworks, including Django, Monorail (.Net), Symfony, CodeIgniter, Yii, CakePHP, Model-Glue, Grails, Catalyst, Seam Framework, Spring Roo, ASP.NET Dynamic Data and ASP.NET MVC Framework's Metadata Template Helpers.

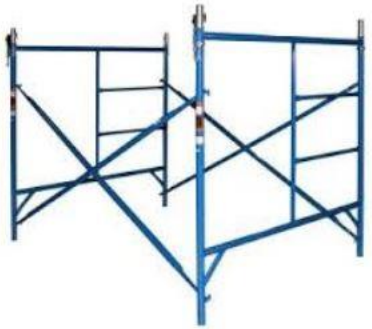
…?

http://blog.saltfactory.net/speedly-developing-web-service-via-scaffolding/

멋쟁이사자처럼 @연세대 김연진

01 Scaffold



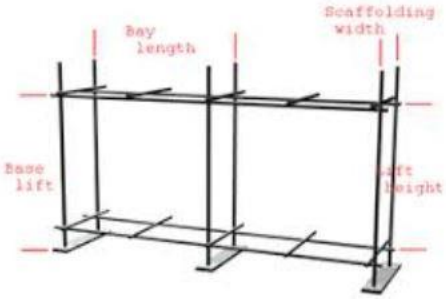


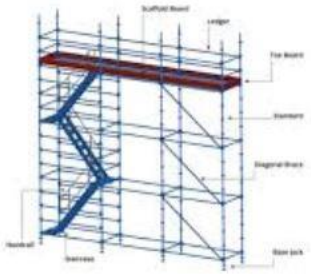
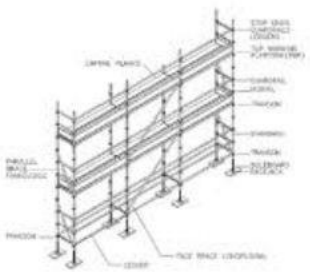










http://blog.saltfactory.net/speedly-developing-web-service-via-scaffolding/

01 Scaffold

1. 그게 몰까?

Scaffolding is a technique supported by some model-view-controller frameworks, in which the programmer may write a specification that describes how the application database may be used. The compiler uses this specification to generate code that the application can use to create, read, update and delete database entries, effectively treating the template as a "scaffold" on which to build a more powerful application. Scaffolding is an evolution of database code generators from earlier development environments, such as Oracle's CASE Generator, and many other 4GL client-server software development products. Scaffolding was popularized by the Ruby on Rails framework. It has been adapted to other software frameworks, including Django, Monorail (.Net), Symfony, CodeIgniter, Yii, CakePHP, Model-Glue, Grails, Catalyst, Seam Framework, Spring Roo, ASP.NET Dynamic Data and ASP.NET MVC Framework's Metadata Template Helpers.

게시판 기능을 한방에 만들어주는 도구

(MVC 모델 기반으로 어플리케이션을 만들 때 써야 하는 코드를 대신 짜준다!)

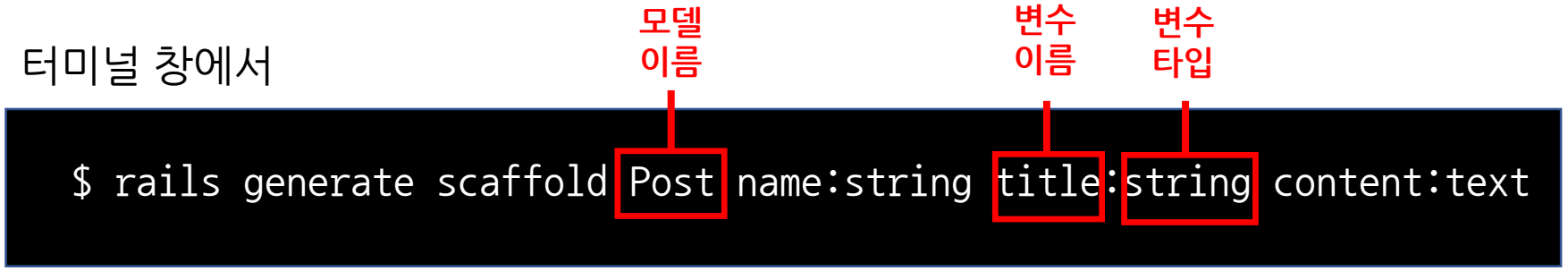
http://blog.saltfactory.net/speedly-developing-web-service-via-scaffolding/

01 Scaffold

2. 활용

▌세팅

터미널 창에서



rails g scaffold [모델이름] [속성]:[타입] [속성]:[타입] 형태
Post라는 모델을 만들고 name 속성은 string 타입, title 속성은 string 타입으로 …
오타나면 안 됨 주의

루비가 최신 버전인 경우

```
$ rails db:migrate
```

루비가 최신 버전이 아닌 경우 (c9의 default 설정)

```
$ rake db:migrate
```

https://rubykr.github.io/rails_guides/getting_started.html#Getting-Up-and-Running-Quickly-with-Scaffolding

01 Scaffold

너무 편하다..

- RESTful 방식으로 CRUD 기능이 있는 코드를 생성
- 하지만 실제 개발 시에는 scaffold 사용을 자제하는 것이 좋다.
  (수정하다보면 나중에 시간이 더 걸린다고 함..)

01 Scaffold

2. 활용

▌세팅

config > routes 에서 아래 내용 추가

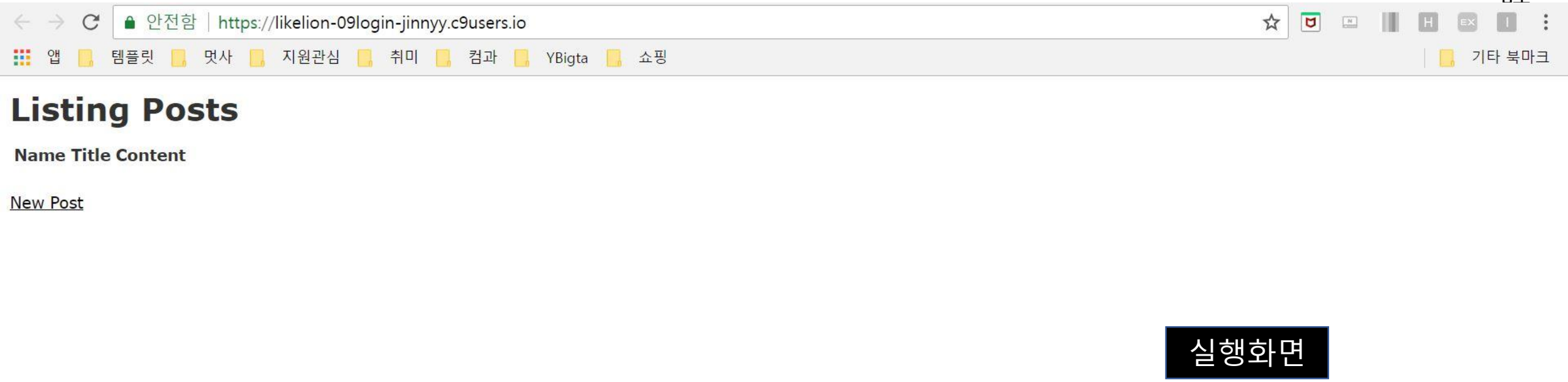
```
root 'posts#index'
```


https://rubykr.github.io/rails_guides/getting_started.html#Getting-Up-and-Running-Quickly-with-Scaffolding

01 Scaffold

2. 활용

멋쟁이사자처럼



실행화면

https://rubykr.github.io/rails_guides/getting_started.html#Getting-Up-and-Running-Quickly-with-Scaffolding

devise

02 Devise

1. 환경구축

▌Gem 설치

Gemfile로 가서 아래 내용 추가

```
gem ‘devise’
```

이후 **터미널 창**에서

```
$ bundle install
```

https://github.com/plataformatec/devise

02 Devise

1. 환경구축

▌모델 생성 및 실행

터미널 창에서

```
$ rails generate devise:install
```

generator를 실행

https://github.com/plataformatec/devise

02 Devise

1. 환경구축

▌모델 생성 및 실행

터미널 창에서

```
$ rails generate devise USER
```

User 모델을 생성한다

루비가 최신 버전인 경우

```
$ rails db:migrate
```

루비가 최신 버전이 아닌 경우

```
$ rake db:migrate
```

https://github.com/plataformatec/devise

02 Devise

2. 활용

▌필터

app > controllers > concerns > **action_controller**에 아래를 입력

```
before_action :authenticate_user!
```

다른 코드가 실행되기 전에 유효한 유저인지 확인한다

https://github.com/plataformatec/devise
http://guides.rubyonrails.org/action_controller_overview.html

멋쟁이사자처럼 @연세대 김연진

02 Devise

2. 활용

▌페이지 접근

로그인 페이지

```
/users/sign_in
```

회원가입 페이지

```
/users/sign_up
```

➔ 이 링크들을 필요한 곳에 연결하면 되겠다!

https://github.com/plataformatec/devise

[M] /README.md × index.html.erb ×

```
1  <p id="notice"><%= notice %></p>
2
3  <a href="/users/sign_in">log in</a>
4
5  <h1>Listing Posts</h1>
6
7  <table>
8    <thead>
9      <tr>
10       <th>Name</th>
11       <th>Title</th>
12       <th>Content</th>
13       <th colspan="3"></th>
14     </tr>
15   </thead>
16
17   <tbody>
18     <% @posts.each do |post| %>
19       <tr>
20         <td><%= post.name %></td>
21         <td><%= post.title %></td>
22         <td><%= post.content %></td>
23         <td><%= link_to 'Show', post %></td>
24         <td><%= link_to 'Edit', edit_post_path(post) %></td>
25         <td><%= link_to 'Destroy', post, method: :delete, data: { confirm: 'Are you sure?' } %></td>
26       </tr>
```

이 부분을 추가!

02 Devise

2. 활용

▌(자주 쓰는) 유용한 **변수**

유저가 로그인했는지 확인

```
user_signed_in?
```

유저의 세션에 접근

```
user_session
```

현재 유저가 누구인지 확인

```
current_user
```

https://github.com/plataformatec/devise
http://guides.rubyonrails.org/action_controller_overview.html

02 Devise

2. 활용

▌(자주 쓰는) 유용한 **변수** - 사용방법

유저가 로그인했는지 확인

```
user_signed_in?
```

view 파일에 아래를 입력

```
<% if user_signed_in? %>        # (아무) 유저가 로그인 한 상태가 true이면
  <%= @posts.each do |post| %>  # controller에서 @posts 변수에 담은 내용에 대해
    <tr>
      <td><%= post.title %></td>
      <td><%= post.content %></td>
    </tr>
  <%= end %>
<% end %>
```

다른 코드가 실행되기 전에 유효한 유저인지 확인한다

https://github.com/plataformatec/devise
http://guides.rubyonrails.org/action_controller_overview.html

02 Devise

2. 활용

▌view 화면 수정하기

터미널 창에서

```
$ rails generate devise:views
```

devise와 관련된 view 파일들을 보이게 하라 (컨트롤러는 views 대신 controllers 입력)

config/initializers/devise.rb에서 아래와 같이 수정

```
config.scoped_views = true
```

https://github.com/plataformatec/devise
http://guides.rubyonrails.org/action_controller_overview.html

멋쟁이사자처럼 @연세대 김연진