

Izone cafe

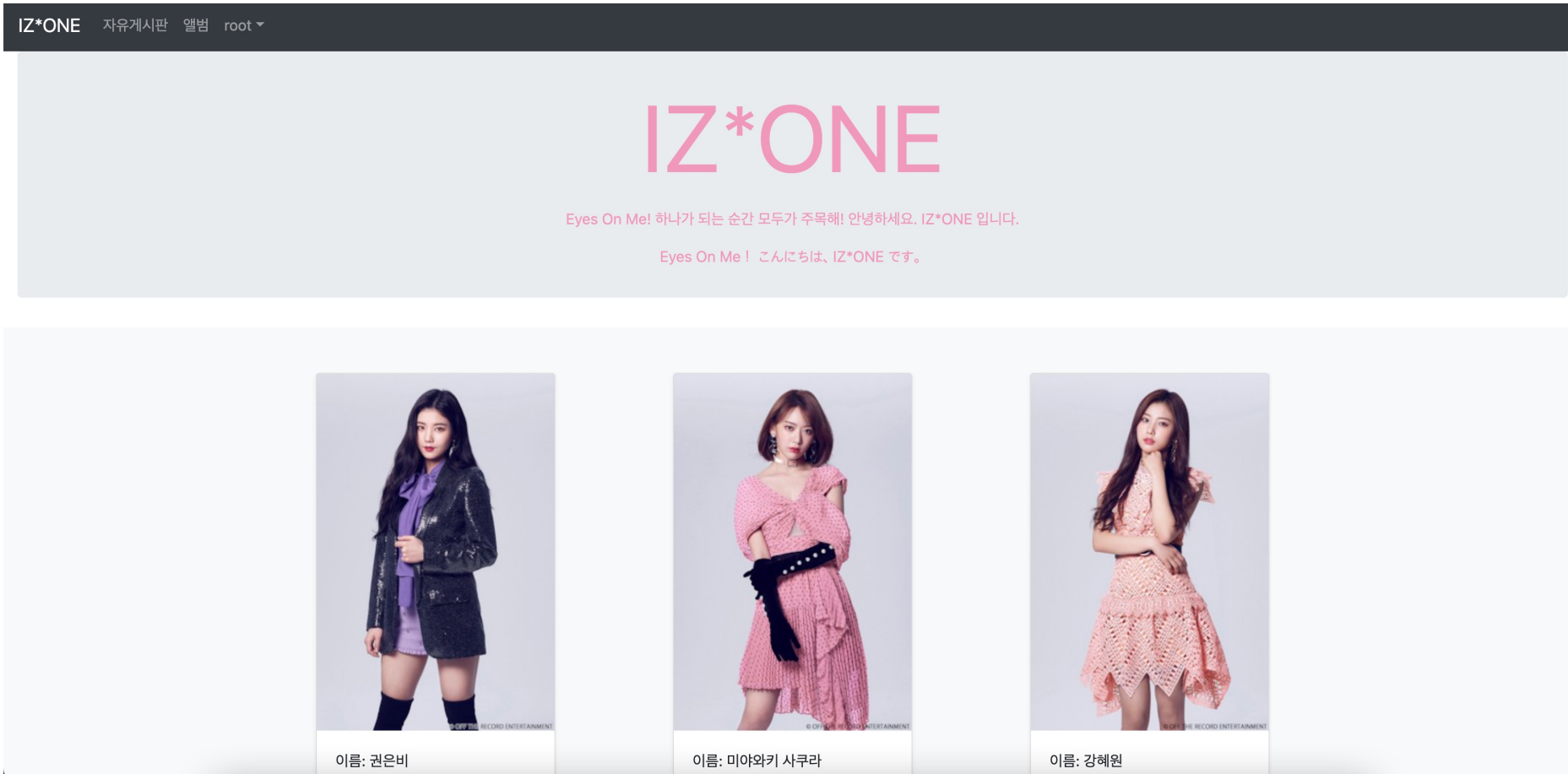
# 제작동기

Django를 접하고 토이 프로젝트를 제작하면서 실력을 기르며, 제대로 된 프로젝트를 제작하고 싶었습니다. 이에 블로그, 카페 등 커뮤니티를 제작하려 했습니다. 그 이유는 DB, 로그인, 게시판 등 다양한 기능을 만들어야 했기 때문입니다. 그렇게 izeone을 소개하는 사이트를 제작했습니다.

# 전체적인 사이트 구조

- 이 사이트의 디자인은 빠른 제작을 위해 부트스트랩으로 제작했습니다.
- 사이트 자체 로그인도 가능하지만, 카카오톡, 페이스북으로 소셜 로그인 기능을 구현했습니다.
- 앨범 탭은 izon 공식 홈페이지에서 크롤링해서 보여주는 탭입니다. (현재는 izon 홈페이지가 폐쇄했기에 작동하지 않는 기능)

# Landing page



멤버들의 프로필 정보를  
카드 형식으로 디자인하여  
12명이나 되는 멤버들을  
직관적으로 볼 수 있도록  
디자인하였습니다.

# 게시판

IZ\*ONE 자유게시판 앨범 root ▾

글쓰기

1

번호	제목	작성자	날짜	조회수
24	권은비	root	2019년 2월 20일 7:24 오후	17
25	권은비	root	2019년 2월 20일 7:25 오후	83
26	12345678	root	2019년 2월 20일 9:00 오후	11
27	1234	root	2019년 2월 21일 9:41 오후	1
30	test	root	2019년 2월 25일 12:10 오후	1
31	tessa	root	2019년 2월 25일 12:10 오후	0
32	gsda	root	2019년 2월 25일 12:10 오후	5
33	123	root	2019년 2월 25일 12:18 오후	2

게시글 번호, 제목, 작성자, 작성 날짜, 조회수 등 기본적인 게시판에 필요한 기능들을 넣었습니다.

```
@login_required(login_url="/login")
```

하지만, Django의 login\_required의 기능을 이용하여 로그인 하지 않는 사람들은 접속하지 못하도록 설정했습니다.

# 회원가입

IZ\*ONE 자유게시판 앨범

### Sign In

아이디

ID 6글자 이상

이메일

Ex) example@abc.com

비밀번호

10글자 이상

비밀번호 확인

비밀번호 확인

SIGN UP

해당 양식과 맞지 않으면  
회원가입이 되지 않게  
설정하였습니다.

# 회원가입

```
class JoinForm(UserCreationForm):
    username = forms.CharField(required = True)
    email = forms.EmailField(required = True)

    class Meta:
        model = User
        fields = ('username', 'email', 'password1', 'password2')

    def save(self, commit=True):
        user = super(JoinForm, self).save(commit=True)
        user.email = self.cleaned_data['email']
        if commit:
            user.save()
        return user
```

forms.py의 회원가입 코드입니다.

HTML <input> 태그에서 forms.py로  
데이터가 보내지며, email 중복 확인  
후 DB에 저장 됩니다.

# 로그인

IZ\*ONE 자유게시판 앨범 갤러리



### Sign In

아이디

비밀번호

**SIGN IN**

**SIGN UP**

  카카오계정으로 로그인

로그인 페이지서는 사이트 자체적으로 회원가입이 가능하며, 추가적으로 페이스북, 카카오톡으로 로그인 가능합니다.



# 로그인

```
<form method="POST">
  {% csrf_token %}
  <div class="form-label-group">
    <p>
      아이디
    </p>
    <input type="text" class="form-control" placeholder="ID" required autofocus {{form.username}} <p>
  </p>
  <p>
    비밀번호
  </p>
  <input type="password" class="form-control" placeholder="Password" required autofocus {{form.password}} <div>
</div>
```

회원가입과 마찬가지로  
forms.py로 데이터를 보내며,  
데이터 검증 후 로그인하는  
코드입니다.

# 소셜 로그인

Django 관리

환영합니다, **ROOT**. [사이트 보기](#) / [비밀번호 변경](#) / [로그아웃](#)

홈 > 소셜 계정 > 소셜 계정

HELLO

Posts + 추가

Users + 추가

계정

이메일 주소 + 추가

사이트

사이트들 + 추가

소셜 계정

소셜 계정 + 추가

소셜 어플리케이션 + 추가

소셜 어플리케이션 토큰 + 추가

인증 및 권한

그룹 + 추가

사용자(들) + 추가

변경할 소셜 계정 선택

Q

검색

액션:  실행 2 중 아무것도 선택되지 않았습니다.

<input type="checkbox"/>	USER	UID	제공자
<input type="checkbox"/>	user15	1034317645	Kakao
<input type="checkbox"/>	user	2268596030130111	Facebook

2 소셜 계정

필터

제공자 (으)로

모두

Facebook

Kakao

소셜 계정 추가 +

소셜 로그인에 성공하면, 다음과 같이 소셜 계정이 생성이 되며 토큰도 발급이 됩니다.

<input type="checkbox"/>	APP	ACCOUNT	TOKEN
<input type="checkbox"/>	kakao	user15	d2bUtBB5u6lluqdl6MBS08qxBLjvFGfvU2Fc2Qop...(truncated)
<input type="checkbox"/>	Facebook	user	EAAYZAKsowzXgBA07ZAbEqrI8Q3Fc5vnaLp19YN2...(truncated)

# 글쓰기

IZ\*ONE 자유게시판 앨범

제목

내용

file

저장

# 글쓰기

```
class PostForm(forms.ModelForm):
    title = forms.CharField()
    text = forms.CharField(widget=forms.Textarea(
        attrs={
            'placeholder': '내용',
            "rows":8,}
    ))

    class Meta:
        model = Post
        fields = ['title', 'text','image']

    def __init__(self, *args, **kwargs):
        super(PostForm, self).__init__(*args, **kwargs)
        self.fields['image'].required = False
```

forms.py의 글쓰기 코드입니다. HTML의  
<input> 태그에서 받은 값으로  
검증하는데, 내용은 8줄 이상 넘어가면  
글을 쓰지 못하도록 설정하였습니다

# 글 수정 및 삭제

IZ\*ONE 자유게시판 앨범

test

- root님 -

수정

삭제

test

0 Comments gailcsauce Disqus' Privacy Policy

1 Login ▾

Favorite

Tweet

Share

Sort by Best ▾



Start the discussion...

LOG IN WITH



OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

Subscribe

Add Disqus to your site

Do Not Sell My Data

DISQUS

# 글 수정

```
@login_required(login_url="/login")
def edit(request, pk):
    post = get_object_or_404(Post , pk=pk)
    if request.method == "POST":
        form = PostForm(request.POST, request.FILES, instance =post)
        if form.is_valid():
            if post.user == User.objects.get(username = request.user.get_username()):
                form.save()
            else:
                return redirect('/')
        return redirect('post')
    else:
        form = PostForm(instance = post)
    context = {
        'form':form,
        'post':post,
    }
    return render(request, 'post/edit.html', context)
```

pk값이 해당 게시물 pk와 일치하지 않으면, 404에러 발생시킵니다.

글쓰기와 비슷하지만, 해당 게시글의 내용을 가져와야 하는 것에서 차이가 있습니다.

# 글 삭제

```
@login_required(login_url="/login")
def delete(request,pk):    #게시글 삭제
    post = Post.objects.get(pk=pk)
    if post.user == User.objects.get(username = request.user.get_username()):
        post.delete()
        return redirect('post')
    else:
        return redirect('/')
```

pk 값이 해당 게시글과 pk값이  
맞는지 확인과, 그 유저가 작성한  
pk가 맞는지 확인 후 삭제가  
됩니다.

# 느낀점

부트스트랩이 빠르게 프론트 엔드를 작성 할 수 있다고만 들었었는데, 직접 사용해 보니 정말 편리하고 빠르게 프론트 엔드를 만들어 갔었으며, Django를 사용해서 게시판 기능을 제작해 나아가면서 Django의 전반적인 기능을 배웠습니다.