ページを１つ追加する手順

１. urls.pyにpathを追加

例：path(r'post/<int:post\_id>/', views.post\_detail, name='post\_detail'),

↑第一引数：適当に他と被らないように(ユーザごとに表示を変えたい場合<int:user\_id>、匿名質問ごとに表示を変えたい場合<int:question\_id>などをつける)

↑第二引数：「views.関数名」でviews.pyの関数に飛ぶ

↑第三引数：name='文字列'でpathの名前を決める

2. views.pyに関数を追加

1の第二引数で決めた関数名で関数を定義

# 第一引数にはrequestを入れる。1の第一引数で<int:post\_id>などを指定した場合は

# 第二引数にpost\_idなどを書く

def post\_detail(request, post\_id):

# 必要なオブジェクトを取得して適当な変数に入れる(この場合、投稿ごとに表示を変えたいから第二引数のpost\_idを利用して一つの投稿を取得)

post = Post.objects.get(pk=post\_id)

# フォームが必要なページの場合、適当な変数にforms.pyにあるformを代入する(この場合CommentForm)(forms.pyのクラス名を書くと呼び出せる)

comme\_form = forms.CommentForm(request.POST or None) # フォームのボタンが押された場合入力されたフォームが入る。押されてない場合、空のフォームが入る。

# 画像のアップロードがある場合は(QuestionForm, ReplyForm, PostForm)、第二引数にrequest.FILES or None をつける

# フォームのボタンが押された場合フォーム.is\_valid()でバリデーションしてくれる。(getメソッドの時は実行されない)

if comme\_form.is\_valid():

# フォーム.save()でフォームの内容をデータベースに保存

comme\_form.save()

# 無事保存できたら return redirect('groomings:「pathの名前」') でそのurlへ飛ぶ

return redirect('groomings:top')

# post.post\_comment.all()でその投稿についてるコメントを取得(model\_explanation.txt 14行目参照)

comments = post.post\_comment.all()

# フォームがvalidじゃなかった場合、もしくはフォームのボタンが押されてない場合(getメソッドの場合)、そもそもフォームのないページの場合などはreturn renderでhtmlを表示する

# 第一引数にrequest、第二引数にhtmlへのpath、第三引数に必要なオブジェクト(フォームとか)を辞書形で渡す

return render(request, 'groomings/post\_detail.html', context={'post': post, 'form': comme\_form, 'comments': comments})

3. templates/groomingsの下に2のrenderで書いたhtmlを作る

親html(bootstrapの設定とか<header>とか書いてる)を継承

{% extends 'groomings/base.html' %}

{block title %}ここにタイトル(<head>の中の<title>)をかく{% endblock %}

{% block title %}投稿詳細ページ{% endblock %} <!-- タイトル -->

{% block content %}ここが本文(<body>の中){% endblock %}

{% block content %}

# 2のrenderの第三引数で書いたオブジェクトが使える

# {{ オブジェクト.カラム名 }} でそのオブジェクトのカラムの内容を取得(models.py)(詳しくはmodel\_explanation.txt)

<h1>{{ post.title }}</h1>

<p>{{ post.text }}</p>

<h2>コメントする</h2>

<form method="POST">　# 画像のアップロードがある場合は(ReplyForm, QuestionForm, PostForm)、enctype="multipart/form-data" も付ける

{% csrf\_token %} # CSRF検証

# 2で辞書形で渡したフォームは{{ 渡されたフォーム.as\_p }}などと書くと自動でフォームを生成してくれる

{{ form.as\_p }}

# 送信ボタンだけは自分で書く

<input type="submit" value="送信">

</form>

<h3>コメント一覧</h3>

<ul>

# {% if ~~ %}{% elif ~~ %}{% else %}などと書くとif文で表示を変えられる(最後に{% endif %}と書くのを忘れない)

# {% if user.is\_authenticated %}と書くとログインしてる人だけの表示にできる

# {% for ○○ in 「2の辞書形で渡した複数インスタンスの入ったオブジェクト」 %}と書くと繰り返し処理ができる(ループの終わりで{% endfor %}書くのを忘れない)

{% for comment in comments %}

<li>{{ comment.user.username }}｜{{ comment.text }}</li>

{% endfor %}

</ul>

{% endblock %}

---------------------------------------------------------------------------------------------------------------------------

フォームが必要なページ

ログインページ, ユーザー登録ページ, ユーザー情報編集ページ,

投稿ページ, 匿名質問をするためのページ(QuestionForm), 投稿詳細ページ(CommentForm), 自分がしたかされた匿名質問の詳細ページ(ReplyForm)

フォームが必要じゃないページ

ユーザーのお気に入り投稿ページ, ユーザー個人ページ, ランキング用のページ, 匿名質問に関するページ(自分にきた質問と自分がした質問を一覧表示)