장고 튜토리얼 정리

manage.py

manage.py - 장고의 모든 명령이 이루어짐

settings.py

settings.py - 장고 프로젝트의 모든 피드와 설정 정보를 담고 있는 파일. 각종 인증 설정 정보까지 포함. e-mail 정보 등

`settings.py 16번째 줄`

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

__file__은 현재 settings.py가 씌여지고 있는 파일을 의미. dirname은 그의 부모 디렉토리를 의미하므로, BASE_DIR은 sttings.py의 부모 디렉토리의 부모 디렉토리인 djangogirls 디렉토리의 절대경로를 의미.

blog/url.py

url.py가 프로젝트 url에 몰려있게 되면, 관리적인 측면에서 좋지 않음. bolg.py를 만들고 include를 이용함

blog/view.py

post_detail

```
def post_detail(request, pk):
    post = Post.objects.get(pk=pk)
```

지정된 pk에 post가 없으면, Post.DoesNotExist 오류가 발생하게 된다. 이 오류를 잡아주지 않으면, 500에러(서버오류)가 발생하게 된다. 지정된 pk에 post가 없다는 것이 서버오류로 처리 되는 것은 적절하지 않다. 따라서 이를, 아래처럼 try except문으로 처리해야 한다.

```
def post_detail(request, pk):
    try:
        post = Post.objects.get(pk=pk)
    except Post.DoesNotExist:
        raise Http404      #django.http.Http404 에 있음
```

그러나, 매번 이렇게 하는 것은 번거로운데, 이를 한번에 처리해주는 메쏘드가 get_object_or_404 이다. 위의 4줄이 아래처럼 한번에 처리된다.

```
def post_detail(request, pk):
    post = get_object_or_404(Post, pk=pk)
```

def post_view(request):

장고에서 form 클래스를 처리하는 view 코드는 거의 모양이 비슷하다. 하나의 덩어리로 익혀두는 것이 좋다.

```
def post_view(request):
    if request.method == "POST":
        form = PostForm(request.Post, request.FILES)
        if form.is_valid():
            post = form.save(commit=False)
            post.author = request.user
            post.published_date = timezone.now()
            post.save()
            return redirect('blog.views.post_detail', post.pk)
    else:
        form = PostForm()
    return render(request, 'blog/post_edit.html', {'form' : form})


def post_edit(request, pk):
    post = get_object_or_404(Post, pk=pk)
    if request.method == "POST":
        form = PostForm(request.POST, instance=post)
        if form.is_valid():
            post = form.save(commit=False)
            post.author = request.user
            post.published_date = timezone.now()
            post.save()
            return redirect('post_detail', pk=post.pk)
    else:
        form = PostForm(instance=post)
    return render(request, 'blog/post_edit.html', {'form': form})
```

blog/form.py

장고에서 form은 매우 중요하다. 사용자의 입력에 대해서 유효성 검사를 수행하고, 수행된 결과를 db에 잘 전달해주는 편리한 기능.