190520

URI에 name 지정하기

uri에 name을 지정해주어서, uri가 변경되었을 때, 그 uri를 src하는? 애들을 일일이 변경할 필요를 없앰

- 지정하기

```
app_name='boards'

urlpatterns = [
    path('<int:pk>/edit/', views.edit, name='edit'),
]
```

- 사용하기

```
redirect('boards:detail', board.pk)
```

```
<a href='{% url 'boards:detail' board.pk %}' />
```

RESTful 하게 URI 설계하기

본래는 행위는 메소드로만 표현해야 하지만,

Http 공식 지원 메소드는 Get, Post 뿐이기 때문에, 할 수 없이 URI에 행위를 표현하고 분기하는 식으로 구현한다.

[new.html]

```
...
<form action="" method="POST"> # action 이 공란 -> 현재 uri로 다시 요청을 보내줌
...
```

[views.py]

```
def new(request):
    if request.method == 'POST':
        title = request.POST.get('title')
        content = request.POST.get('content')

        board = Board(title=title, content=content)
        board.save()
        return redirect('boards:detail', board.pk)
    else:
        return render(request, 'boards/new.html')
```

외래키를 활용해서 댓글 기능 구현하기

1. 외래키 있는 model 생성

```
class Comment(models.Model):
    content = models.CharField(max_length=20)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    board = models.ForeignKey(Board, on_delete=models.CASCADE) # 외래키

    def __str__(self):
        return f'<Board({self.board_id}): Comment({self.id})>'
```

2. 마이그레이션

```
$ python manage.py makemigrations
$ python manage.py migrate
```

3. SQLite 확인

```
$ sqlite3 db.sqlite3
sqlite> .tables
sqlite> .schema boards_comment
```

4. Comment 추가

```
In [2]: board = Board.objects.get(pk=15)
In [3]: comment = Comment()
In [4]: comment.content = 'this is comment!'
In [5]: comment.board = board
In [7]: comment.save()
```

5. Board-Comment 확인

```
In [10]: board.comment_set.all() # 해당 board의 comment set 받아옴
Out[10]: <QuerySet [<Comment: <Board(15): Comment(1)>>, <Comment: <Board(15): Comment(2)>>]>
```

6. Admin 설정

```
class CommentAdmin(admin.ModelAdmin):
    list_display = ('content', 'created_at', 'updated_at')

admin.site.register(Comment, CommentAdmin)
```

7. 댓글 작성

[views.py]

```
def comments_create(request, board_pk):
    board = Board.objects.get(pk=board_pk)
    if request.method == 'POST':
        # 댓글 작성
        comment = Comment()
        comment.board = board
        comment.content = request.POST.get('content')
        comment.save()
        return redirect('boards:detail', board.pk)
    else:
        return redirect('boards:detail', board.pk)
```

[detail.html]

```
...
    <form action="{% url 'boards:comment_create' board.pk %}" method='POST'>
        {% csrf_token %}
        댓글 달기 : <input type="text" name="content"/>
        <input type="submit" value="Submit"/>
    </form>

    <h3><b>댓글</b></h3>
    <p>
        {% for comment in board.comment_set.all %}
            <li>{{ comment.content }}</li>
        {% endfor %}
    </p>
...
```

8. 댓글 삭제

[views.py]

```
def comment_delete(request, board_pk, comment_pk):
    comment = Comment.objects.get(pk = comment_pk)
    if request.method == 'POST':
        # 댓글 삭제
        comment.delete()
        return redirect('boards:detail', board_pk)
    else:
        return redirect('boards:detail', board_pk)
```

[urls.py]

```
...
    path('<int:board_pk>/comments/<int:comment_pk>/delete/', views.comment_delete, name='comment_delete'),
    path('<int:board_pk>/comments/', views.comments_create, name='comments_create'),
...
```

[detail.html]

```
...
    <p>
        {% for comment in comments %}
            <li>
                {{ comment.content }}
                <form action='{% url 'boards:comment_delete' board.pk comment.pk %}', method='POST' style="display:inline">
                    {% csrf_token %}
                    <button>삭제</button>
                </form>
            </li>
        {% endfor %}
    </p>
...
```

9. 댓글 개수에 따른 view

[detail.html]

```
<h3><b>댓글 ({{ comments | length }}개)</b></h3>
    ...
    <p>
        {% for comment in comments %}
                        ...
        {% empty %}
            아직 댓글이 없습니다.
        {% endfor %}
    </p>
```