

# Django Model Form

# Model Form

---

- Model과 관련된 데이터의 생성, 유효성 검사 및 처리를 제공
- Django에서 제공하는 forms와 작성한 model을 결합하여 빠르고 간편하게 입력폼 생성이 가능

# Model Form

## 1. App에 forms.py 파일을 생성하고, ModelForm class 정의하기

- model을 연결

```
# articles/forms.py
from django import forms
from .models import Article

class ArticleForm(forms.ModelForm):
    class Meta:
        model = Article
        # 모든 필드를 사용
        # fields = "__all__"

        # 지정 필드만 사용
        # fields = ['content']

        # 특정 필드 제외
        # exclude = ['title']

        fields = ['title', 'content']
        widgets = {
            'title': forms.TextInput(attrs={"placeholder": "제목 입력창"}),
            'content': forms.Textarea(attrs={"class": "my-class"}),
        }
        error_messages = {
            'title': {
                'max_length': '입력 길이를 초과했습니다.'
            }
        }
}
```

# Model Form

## 1. App에 forms.py 파일을 생성하고, ModelForm class 정의하기

- Form에서 사용할 필드를 지정

```
# articles/forms.py
from django import forms
from .models import Article

class ArticleForm(forms.ModelForm):
    class Meta:
        model = Article
        # 모든 필드를 사용
        # fields = "__all__"

        # 지정 필드만 사용
        # fields = ['content']

        # 특정 필드 제외
        # exclude = ['title']

        fields = ['title', 'content']
        widgets = {
            'title': forms.TextInput(attrs={"placeholder" : "제목 입력창"}),
            'content': forms.Textarea(attrs={"class" : "my-class"}),
        }
        error_messages = {
            'title': {
                'max_length': '입력 길이를 초과했습니다.'
            }
        }
}
```

# Model Form

## 1. App에 forms.py 파일을 생성하고, ModelForm class 정의하기

- 폼 필드에 적용되는 외관과 동작을 설정 ([Django 공식 홈페이지 참고](#))

```
# articles/forms.py
from django import forms
from .models import Article

class ArticleForm(forms.ModelForm):
    class Meta:
        model = Article
        # 모든 필드를 사용
        # fields = "__all__"

        # 지정 필드만 사용
        # fields = ['content']

        # 특정 필드 제외
        # exclude = ['title']

        fields = ['title', 'content']
        widgets = {
            'title': forms.TextInput(attrs={"placeholder": "제목 입력창"}),
            'content': forms.Textarea(attrs={"class": "my-class"}),
        }
        error_messages = {
            'title': {
                'max_length': '입력 길이를 초과했습니다.'
            }
        }
```

# Model Form

## 1. App에 forms.py 파일을 생성하고, ModelForm class 정의하기

- 필드의 조건에 맞지 않는 경우 출력할 에러메세지 설정

```
# articles/forms.py
from django import forms
from .models import Article

class ArticleForm(forms.ModelForm):
    class Meta:
        model = Article
        # 모든 필드를 사용
        # fields = "__all__"

        # 지정 필드만 사용
        # fields = ['content']

        # 특정 필드 제외
        # exclude = ['title']

        fields = ['title', 'content']
        widgets = {
            'title': forms.TextInput(attrs={"placeholder": "제목 입력창"}),
            'content': forms.Textarea(attrs={"class": "my-class"}),
        }
        error_messages = {
            'title': {
                'max_length': '입력 길이를 초과했습니다.'
            }
        }
```

# Model Form

---

## 2. Model Form을 template에 전달하기 위해서 view 함수 수정

```
from django.shortcuts import render, redirect
from .models import Article
from .forms import ArticleForm

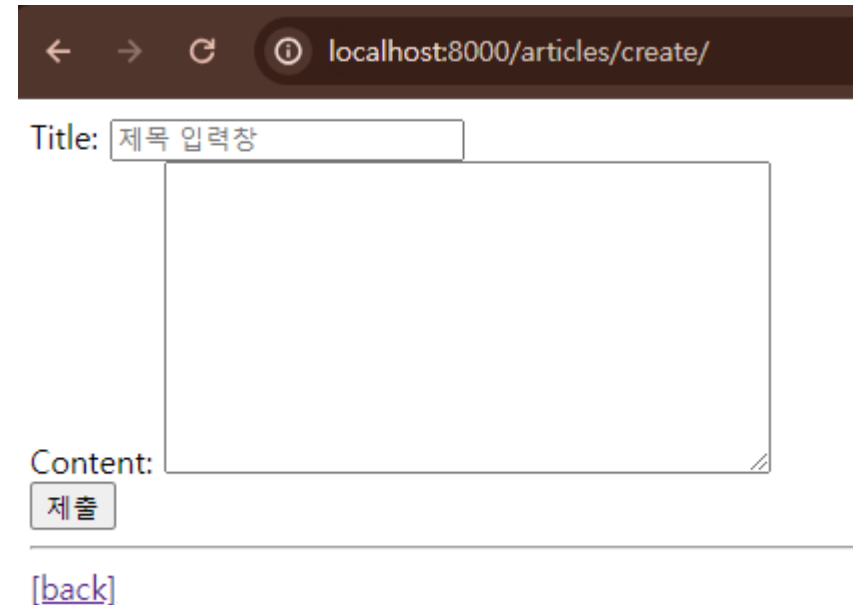
# articles/views.py
def create(request):
    form = ArticleForm()
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```

# Model Form

## 3. Template 에서 ModelForm 사용하기

```
<!-- templates/articles/create.html-->
{% extends "articles/base.html" %}

{% block content %}
    {% comment %} <form action="{% url 'articles:save' %}" method="POST">
        {% csrf_token %}
        <div>
            <label for="title">Title: </label>
            <input type="text" name="title" id="title">
        </div><br>
        <div>
            <label for="content">Content: </label>
            <textarea name="content" id="content"></textarea>
        </div><br>
        <input type="submit">
    </form> {% endcomment %}
    <form action="{% url 'articles:save' %}" method="POST">
        {% csrf_token %}
        {{ form }}
        <input type="submit">
    </form>
    <hr>
    <a href="{% url 'articles:index' %}">[back]</a>
{% endblock content %}
```



← → ↻ ⓘ localhost:8000/articles/create/

Title: 제목 입력창

Content:

제출

[\[back\]](#)



# Model Form

## 4. 생성된 Form을 이용해서 데이터 제출하기

- is\_valid: 모델에 설정한 제약조건과 데이터가 일치하는 지 검사

← → ↻ ⓘ localhost:8000/articles/create/

Title:

영화 콘텐츠

Content:

[\[back\]](#)

```
# articles/views.py
def save(request):
    form = ArticleForm(request.POST)
    if form.is_valid():
        article = form.save()
        return redirect('articles:detail', article.pk)
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```

← → ↻ ⓘ localhost:8000/articles/detail/15/

**DETAIL**

15 번째 글

---

제목: 영화 제목

내용: 영화 콘텐츠

---

[\[수정하기\]](#)

[\[삭제하기\]](#)

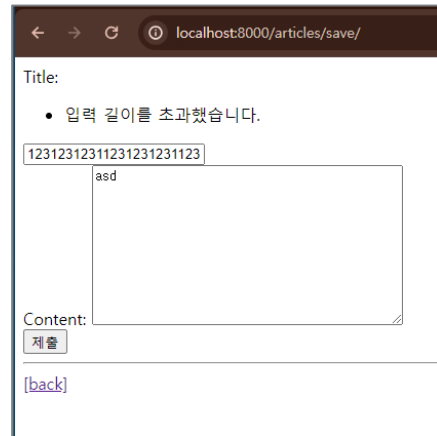
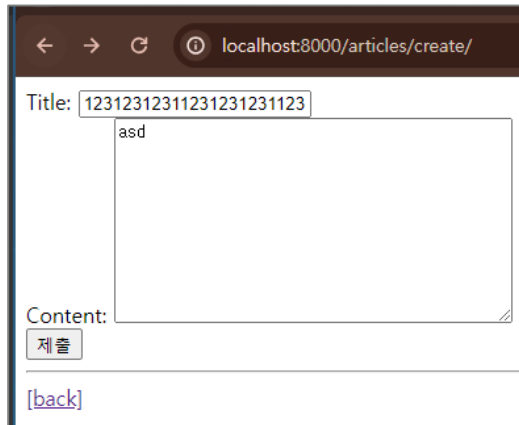
---

[\[back\]](#)

# Model Form

## 4. 생성된 Form을 이용해서 데이터 제출하기

- is\_valid: 모델에 설정한 제약조건과 데이터가 일치하는 지 검사

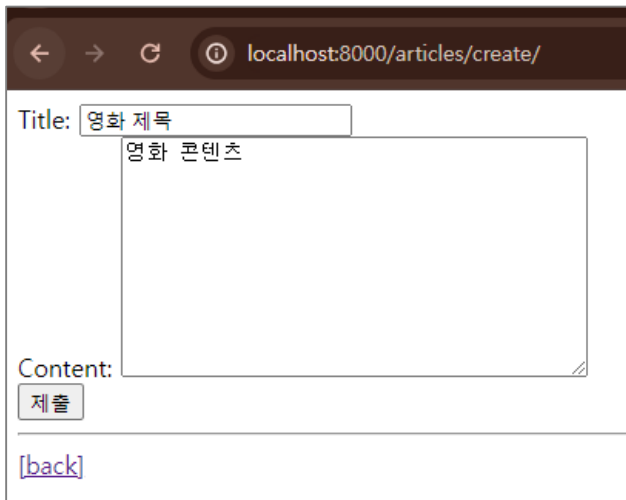


```
error_messages = {  
    'title': {  
        'max_length': '입력 길이를 초과했습니다.',  
    }  
}
```

# Model Form

## 4. 생성된 Form을 이용해서 데이터 제출하기

- save: Form에 입력한 데이터를 기반으로 데이터 생성



← → ↻ ⓘ localhost:8000/articles/create/

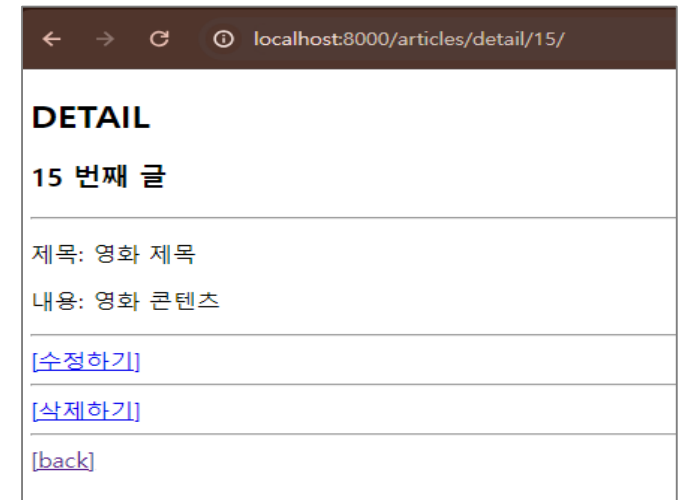
Title: 영화 제목

영화 콘텐츠

Content:

[\[back\]](#)

```
# articles/views.py
def save(request):
    form = ArticleForm(request.POST)
    if form.is_valid():
        article = form.save()
        return redirect('articles:detail', article.pk)
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```



← → ↻ ⓘ localhost:8000/articles/detail/15/

**DETAIL**

15 번째 글

제목: 영화 제목

내용: 영화 콘텐츠

[\[수정하기\]](#)

[\[삭제하기\]](#)

[\[back\]](#)

# View 함수 구조 변화

# View 함수 구조 변화

현재까지 생성한 create 함수와 save 함수의 차이 비교

```
# articles/views.py
def create(request):
    form = ArticleForm()
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```

GET method만 처리

```
# articles/views.py
def save(request):
    form = ArticleForm(request.POST)
    if form.is_valid():
        article = form.save()
        return redirect('articles:detail', article.pk)
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```

POST method만 처리

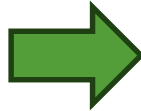
둘 다 데이터를 생성하는 과정인데 하나로 합칠수는 없을까?

# View 함수 구조 변화

## 1. Create 함수와 save 함수의 공통점과 차이점을 기반으로 결합

```
# articles/views.py
def save(request):
    form = ArticleForm(request.POST)
    if form.is_valid():
        article = form.save()
        return redirect('articles:detail', article.pk)
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)

def create(request):
    form = ArticleForm()
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```



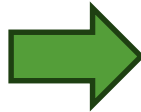
```
# articles/views.py
def create(request):
    if request.method == 'POST':
        form = ArticleForm(request.POST)
        if form.is_valid():
            article = form.save()
            return redirect('articles:detail', article.pk)
    else:
        form = ArticleForm()
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```

# View 함수 구조 변화

## 2. 두 함수의 차이점인 request method를 기준으로 분리

```
# articles/views.py
def save(request):
    form = ArticleForm(request.POST)
    if form.is_valid():
        article = form.save()
        return redirect('articles:detail', article.pk)
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)

def create(request):
    form = ArticleForm()
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```



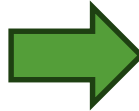
```
# articles/views.py
def create(request):
    if request.method == 'POST':
        form = ArticleForm(request.POST)
        if form.is_valid():
            article = form.save()
            return redirect('articles:detail', article.pk)
    else:
        form = ArticleForm()
        context = {
            'form': form
        }
        return render(request, 'articles/create.html', context)
```

# View 함수 구조 변화

## 3. POST일 경우 데이터 생성 로직 진행

```
# articles/views.py
def save(request):
    form = ArticleForm(request.POST)
    if form.is_valid():
        article = form.save()
        return redirect('articles:detail', article.pk)
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)

def create(request):
    form = ArticleForm()
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```



```
# articles/views.py
def create(request):
    if request.method == 'POST':
        form = ArticleForm(request.POST)
        if form.is_valid():
            article = form.save()
            return redirect('articles:detail', article.pk)
    else:
        form = ArticleForm()
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```

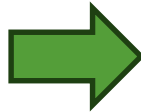


# View 함수 구조 변화

## 4. POST가 아닐 경우 단순 Form으로 인스턴스 생성 후 template 호출

```
# articles/views.py
def save(request):
    form = ArticleForm(request.POST)
    if form.is_valid():
        article = form.save()
        return redirect('articles:detail', article.pk)
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)

def create(request):
    form = ArticleForm()
    context = {
        'form': form
    }
    return render(request, 'articles/create.html', context)
```



```
# articles/views.py
def create(request):
    if request.method == 'POST':
        form = ArticleForm(request.POST)
        if form.is_valid():
            article = form.save()
            return redirect('articles:detail', article.pk)
    else:
        form = ArticleForm()
        context = {
            'form': form
        }
        return render(request, 'articles/create.html', context)
```

# View 함수 구조 변화

사용하지 않는 save 함수를 호출하는 곳 제거

```
<!-- templates/articles/create.html-->
...
{% comment %} <form action="{% url 'articles:save' %}" method="POST"> {% endcomment %}
<form action="{% url 'articles:create' %}" method="POST">
    {% csrf_token %}
    {{ form }}
    <input type="submit">
</form>
```

```
# articles/urls.py
app_name = 'articles'
urlpatterns = [
    # path('save/', views.save, name='save'),
    ...
]
```

# ModelForm Update

---

기존 edit 과 update함수를 합쳐서 구현

```
# articles/views.py
def update(request, pk):
    article = Article.objects.get(pk=pk)
    if request.method == 'POST':
        form = ArticleForm(request.POST, instance=article)
        if form.is_valid():
            form.save()
            return redirect('articles:detail', article.pk)
    else:
        form = ArticleForm(instance=article)
    context = {
        'form': form,
        'article': article
    }
    return render(request, 'articles/edit.html', context)
```

# ModelForm Update

---

기존 edit 과 update함수를 합쳐서 구현

- ModelForm에 instance를 연결해줘서 생성이 아닌 업데이트라고 알려주기

```
# articles/views.py
def update(request, pk):
    article = Article.objects.get(pk=pk)
    if request.method == 'POST':
        form = ArticleForm(request.POST, instance=article)
        if form.is_valid():
            form.save()
            return redirect('articles:detail', article.pk)
    else:
        form = ArticleForm(instance=article)
    context = {
        'form': form,
        'article': article
    }
    return render(request, 'articles/edit.html', context)
```

# ModelForm Update

---

기존 edit 과 update함수를 합쳐서 구현

- method가 GET 일 때, edit 창에 기존 데이터 입력을 위해 instance에 데이터 연결

```
# articles/views.py
def update(request, pk):
    article = Article.objects.get(pk=pk)
    if request.method == 'POST':
        form = ArticleForm(request.POST, instance=article)
        if form.is_valid():
            form.save()
            return redirect('articles:detail', article.pk)
    else:
        form = ArticleForm(instance=article)
    context = {
        'form': form,
        'article': article
    }
    return render(request, 'articles/edit.html', context)
```