

Djagno 기반의 웹프로그래밍

이 소 영

yisy0703@naver.com

Web Frameworks for Python(<https://wiki.python.org/moin/WebFrameworks>)

장고 공식 사이트(<https://www.djangoproject.com/>)

장고 공식 소스 저장소(<http://github.com/django/django>)

장고 참조 문서 영어(<https://docs.djangoproject.com/en/3.2/>)

Agenda

Django(장고) 기반의 파이썬 웹 프로그래밍

Ch01. Django 시작하기

1. Django 란?
2. 개발 환경 구축
3. Django 구조

Ch02. Django App

1. Django Project
2. Model
3. View

Ch03. Model

1. Model 속성 및 옵션
2. Relationship
3. Migrations
4. Admin App

Ch04. Django SQL

1. Django shell
2. Manager & QuerySet
3. 조회 SQL
4. 생성/수정/삭제 SQL
5. Django-Debug-Toolbar

Ch05. Template

1. Template Loader
2. URL Dispatcher
3. Template 상속
4. Template Engines
5. Template Filter

Ch06. Django View

1. View 기본
2. View 활용

Ch07. Django Form

1. HTML form
2. CSRF
3. HttpRequest/HttpResponse
4. Django Form
5. Django Model Form
6. Form Validation

Ch08. File 관리

1. Static Files
2. Media Files
3. Image Thumbnail

Ch09. 사용자 인증

1. Auth App
2. 회원가입 구현
3. 로그인/아웃 구현
4. Oauth 라이브러리 활용

Ch 07 Django Form

1. HTML Form
2. CSRF
3. HttpRequest/HttpResponse
4. Django Form
5. Django Model Form
6. Form Validation

Chapter 07. Django Form

- Cross Site Request Forgery protection
(<https://docs.djangoproject.com/en/3.2/ref/csrf/>)
- MultiValueDict
(<https://github.com/django/django/blob/3.2/django/utils/datastructures.py#L42>)
- QueryDict
(<https://docs.djangoproject.com/en/2.2/ref/request-response/#querydict-objects>)
- Form fields
(<https://docs.djangoproject.com/en/3.2/ref/forms/fields/>)
- ModelForm
(<https://docs.djangoproject.com/en/3.2/topics/forms/modelforms/>)
- Django 폼 처리 과정
(https://developer.mozilla.org/ko/docs/Learn/Server-side/Django/Forms#Django_폼_처리_과정)
- full_clean()
(<https://github.com/django/django/blob/master/django/forms/forms.py#LC368>)
- Validators
(<https://docs.djangoproject.com/en/3.2/ref/validators/>)
- Built-in validators
(<https://docs.djangoproject.com/en/3.2/ref/validators/#built-in-validators>)
- Form.add_error(field, error)
(https://docs.djangoproject.com/en/3.2/ref/forms/api/#django.forms.Form.add_error)



1. HTML Form



HTML FORM

1. HTML Form

- form 태그는 하나 이상의 위젯(Widget)으로 구성
- action : 처리 요청 URL
- method : 처리 요청 방식
 - GET 방식 : Query String이 요청정보 헤더에 포함되어 전달
 - POST 방식 : Query String이 요청정보 바디에 포함되어 전달
- enctype : POST 방식에서만 유효
 - application/x-www-form-urlencoded (default)
 - multipart/form-data : 파일 업로드 가능

```

<form action="queryTest" method="GET">
  ID : <input type="text" name="id" /><br/>
  비밀번호 : <input type="password" name="pwd" /> <br/>
  이름 : <input type="text" name="name" /> <br/>
  취미 :
    <input type="checkbox" name="hobby" value="climbing" /> 등산
    <input type="checkbox" name="hobby" value="traveling" /> 여행 <br/>
  성별 :
    <input type="radio" name="gender" value="male" />남자
    <input type="radio" name="gender" value="female" />여자<br/>
  종교 :
    <select name="religion">
      <option value="Christianity"> 기독교
      <option value="Buddhism"> 불교
    </select> <br/>
  자기소개:<br/>
    <textarea cols="30" rows="10" name="introduction"></textarea> <br/>
  <input type="submit" value="전송" />
  <input type="reset" value="지우기" />
</form>

```

ID :
 비밀번호 :
 이름 :
 취미 : ☐ 등산 ☐ 여행
 성별 : ☐ 남자 ☐ 여자
 종교: 기독교 ▼
 자기소개:

Query String

1. HTML Form

- name=Amy&age=23&picture=amy.jpg
- GET방식

```
request.GET : <QueryDict : {'name':['Amy'],'age':['23'],'picture':['amy.png']}>  
request.POST : <QueryDict : { }>  
request.FILES : <MultiValueDict: { }>
```

- POST 방식

```
request.GET : <QueryDict : { }>  
request.POST : <QueryDict : {'name':['Amy'],'age':['23'],'picture':['amy.png']}>  
request.FILES : <MultiValueDict: { }>
```

- <form method="POST" enctype="multipart/form-data">

```
request.GET:<QueryDict : { }>  
request.POST:<QueryDict : {'name':['Amy'],'age':['23']}>  
request.FILES:<MultiValueDict: {'picture':[<InMemoryUploadFile>'amy.jpg'(image/png)]}>
```




2. CSRF

(Cross Site Request Forgery)



book 앱(1)

2. CSRF

```
D:\src\myproject>python manage.py startapp book
```

```
# book/models.py
```

```
from django.db import models
```

```
class Book(models.Model):
```

```
    title = models.CharField(max_length=50)
```

```
    author = models.CharField(max_length=50)
```

```
    publisher = models.CharField(max_length=50)
```

```
    publication_date = models.DateTimeField(auto_now_add=True)
```

```
# book/views.py
```

```
from django.shortcuts import render, redirect, get_object_or_404
```

```
from django.views.generic import CreateView, ListView
```

```
from .models import Book
```

```
from .forms import BookForm
```

```
book_new = CreateView.as_view(model=Book, fields='__all__')
```

```
book_list = ListView.as_view(model=Book)
```

book 앱(2)

2. CSRF

```
# book/urls.py
from django.urls import path
from . import views
app_name = 'book'
urlpatterns = [
    path('', views.book_list, name='list'),
    path('new/', views.book_new, name='new'),
    path('<id>/edit/', views.book_edit, name='edit'),
]
# book/urls.py
admin.site.register(Book)
```

book 앱(3)

2. CSRF

```
{% extends 'layout.html' %}
{% block content %}
    <h2>Book List</h2>
    {% if book_list %}
        <ul>
            {% for book in book_list%}
                <li>
                    {{book.title}}/{{book.author}} /{{book.publisher}}
                    <a href="{% url 'book:edit' book.id %}">[수정]</a>
                </li>
            {% endfor %}
        </ul>
    {% else %}
        <p>목록이 없습니다.</p>
    {% endif %}

    <a href="{% url 'book:new' %}">[BOOK 추가]</a>
{% endblock %}
```

book 앱(4)

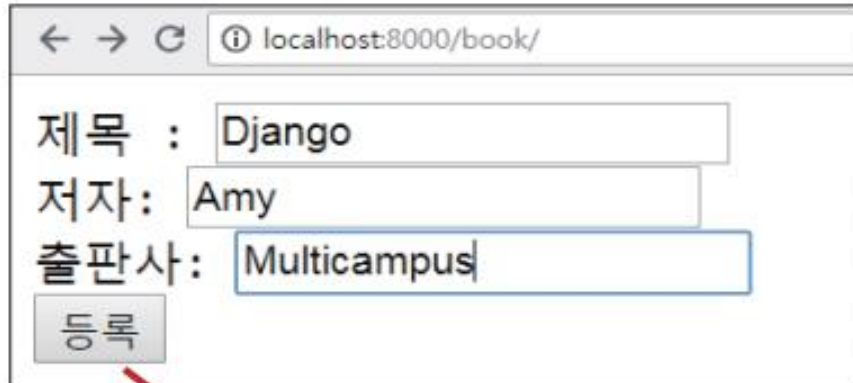
2. CSRF

```
# book/templates/book/book_form.html
<html>
<head>
<title>책 등록</title>
</head>
<body>
  <form action="" method="post">
    제목 : <input type="text" name="title"><br>
    저자 : <input type="text" name="author"><br>
    출판사: <input type="text" name="publisher"><br>
    <input type="submit" value="등록">
  </form>

</body>
</html>
```

book 앱(5)

2. CSRF



제목 : Django
저자: Amy
출판사: Multicampus
등록



CSRF

2. CSRF

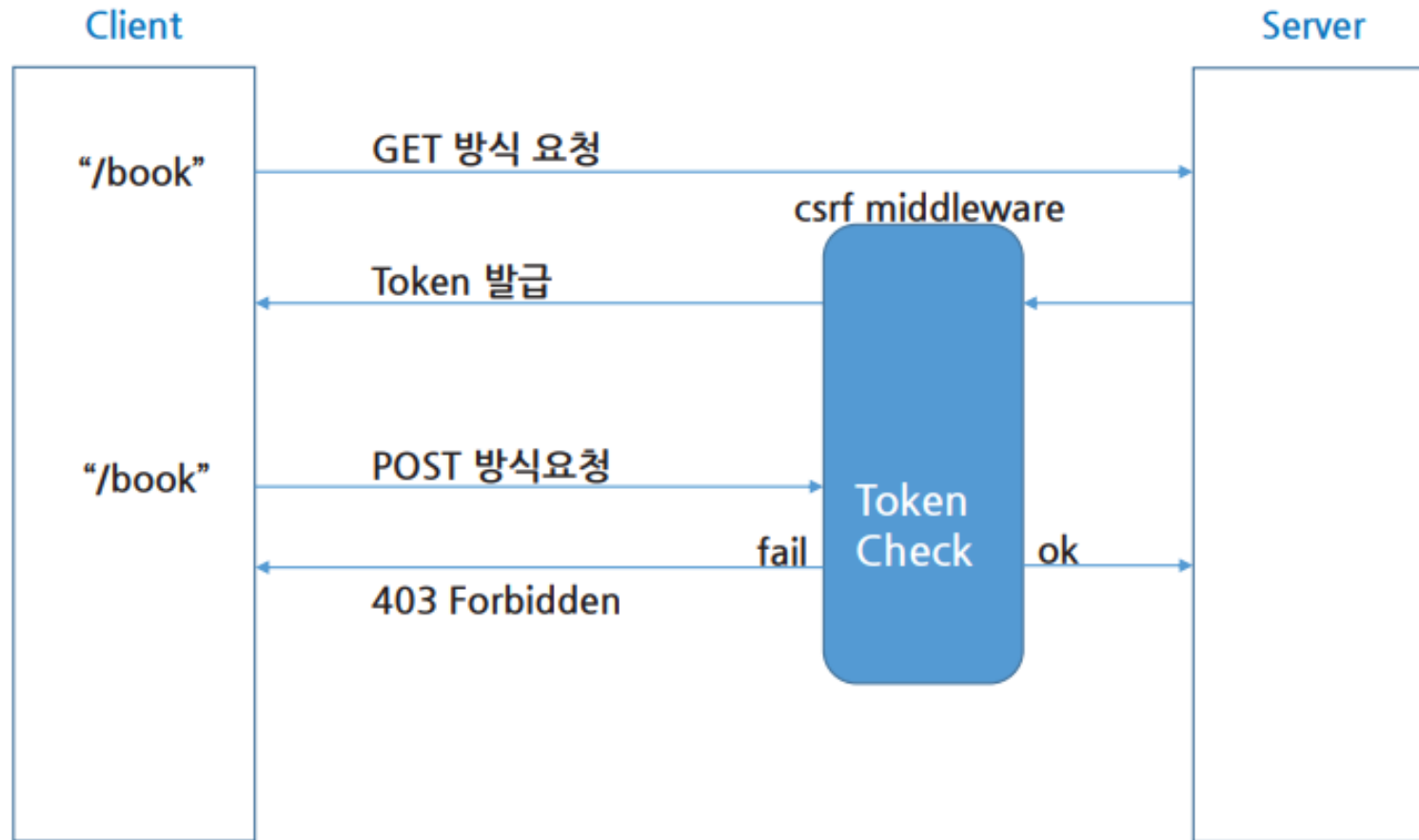
- Cross-Site request forgery(사이트 간 요청 위조 공격)
- 사용자가 의도하지 않은 글쓰기, 쇼핑 공격

```
<body onload="attack()">
  <form action="B" method="post">
    <input type="hidden" name="title" value="스팸 제목" />
    <input type="hidden" name="content" value="스팸 내용" />
  </form>
</body>
```

- CsrfViewMiddleware
 - GET 요청시 csrf token 발급
 - POST 요청시 csrf token 체크
 - token 체크 오류시 403 Forbidden 응답
- csrf는 현재 요청이 유효한지 여부만 판단

CSRF

2. CSRF



CSRF

2. CSRF

```
# book/templates/book/book_form.html
<html>
<head>
<title>책 등록</title>
</head>
<body>
  <form action="" method="post">
    {% csrf_token %}
    제목 : <input type="text" name="title"><br>
    저자 : <input type="text" name="author"><br>
    출판사: <input type="text" name="publisher"><br>
    <input type="submit" value="등록">
  </form>
</body>
</html>
```

```
<html>
  <head>
    <title>책 등록</title>
  </head>
  <body>
    <form action="" method="post">
      <input type="hidden" name="csrfmiddlewaretoken" value="wcrAW0x0R3dsg2bevIL52RZThF91kUpYdJsByRtcfbpS7wuyUuHM1Ne0iz0qmat0" />
      제목 : <input type="text" name="title"><br>
      저자 : <input type="text" name="author"><br>
      출판사: <input type="text" name="publisher"><br>
      <input type="submit" value="등록">
    </form>
  </body>
</html>
```



3. HttpRequest/HttpResponse



HttpRequest 소스(https://docs.djangoproject.com/en/3.2/_modules/django/http/request/#HttpRequest)
HttpRequest 객체(<https://docs.djangoproject.com/en/3.2/ref/request-response/#httprequest-objects>)
HttpResponse 소스(<https://github.com/django/django/blob/3.2/django/http/response.py>)
HttpResponse 객체(<https://docs.djangoproject.com/en/3.2/ref/request-response/#httppresponse-objects>)

HttpRequest objects

3. HttpRequest / HttpResponse

- 클라이언트로부터 전송된 요청정보를 처리하는 객체
- View의 첫번째 인자로 전달됨
- Form 처리 관련 속성
 - `request.method` : "GET" 또는 "POST" 요청방식 정보를 가짐.
 - `request.GET` : GET방식으로 전달된 질의 문자열 정보를 갖는 속성. 타입은 QueryDict 임
 - `request.POST` : POST방식으로 전달된 질의 문자열 정보를 갖는 속성. 타입은 QueryDict 임
 - `request.FILES` : POST방식으로 업로드된 업로드 파일에 대한 정보를 갖는 속성. 타입은 MultiValueDic 임

HttpRequest objects

3. HttpRequest / HttpResponse

● MultiValueDict

- 사전(Dict)을 상속함
- 기본적인 Dict은 key의 중복을 허용하지 않음.
- MultiValueDict은 질의 문자열의 정보를 갖는 객체로서 동일 key를 허용함
예) corlor=red&color=blue&color=yellow

● QueryDict

- MultiValueDict을 상속
- 수정 불가능한(Immutable) MultiValueDict
- Dict와 MultiValueDict은 수정이 가능함

MultiValueDict

3. HttpRequest / HttpResponse

```
from django.utils.datastructures import MultiValueDict

d = MultiValueDict({'name': ['Amy', 'Tobey'], 'position': ['Developer']})
d['name']          # 파이썬의 사전처럼 하나의 값만 나옴
'Tobey'
d.getlist('name')  # name 키값을 가진 모든 값을 조회
['Amy', 'Tobey']
d.getlist('age')
[]
d['name'] = 'Josh'
d
<MultiValueDict: {'name': ['Josh'], 'position': ['Developer']}>
```

QueryDict

3. HttpRequest / HttpResponse

```
from django.http import QueryDict

qd = QueryDict('name=Amy&name=Tobey&position=Developer', encoding='utf8')
qd['name']
'Tobey'
qd.getlist('name')
['Amy', 'Tobey']

qd['name'] = 'Josh'    #오류 발생
```

HttpResponse

3. HttpRequest / HttpResponse

- 응답정보 처리, View 함수의 리턴값으로 전달
- HttpResponse 생성 함수
 - `django.shortcuts.render`
 - `django.shortcuts.redirect`
 - `django.http.JsonResponse`



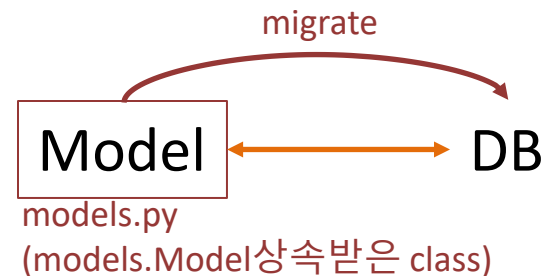
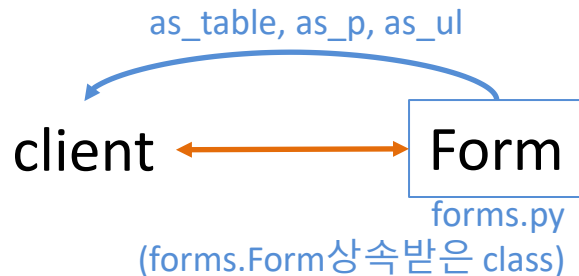
4. Django Form



Django Form

4. Django Form

- Model 클래스와 비슷하게 Form 클래스 정의
- 기능
 - 입력폼 HTML 생성 : `as_table()`, `as_p()`, `as_ul()`
`<tr>` `<p>` ``
 - 입력폼의 값들을 검증(validation) 및 값 변경
 - 검증을 완료한 값들을 사전 타입으로 제공(cleaned_data)



Django Form 처리 방식

4. Django Form

- Django는 동일한 요청에 대해 GET/POST를 구분하여 서로 다른 작업을 함
- GET 방식으로 요청 시 : 입력폼 출력
- POST 방식으로 요청 시
 - 데이터의 유효성을 검증
 - 검증 성공시 : 입력 데이터 저장후 success URL로 이동
 - 검증 실패시 : 오류메시지와 함께 입력폼으로 이동

Form Fields

4. Django Form

- <https://docs.djangoproject.com/en/3.2/ref/forms/fields>
- **Model Fields 와 유사**
 - Model Field : Database Field와 매핑
 - Form Field : HTML Form Field와 매핑
- BooleanField, CharField, ChoiceField, DateField, DateTimeField, EmailField, FileField, ImageField, FloatField, IntegerField, RegexField 등

```
# form.py
from django import forms
from .models import Book

def min_length_3_validator(value):
    if len(value) < 3:
        raise forms.ValidationError('3글자 이상 입력해주세요.')

class BookForm(forms.Form):
    title = forms.CharField(label="제목")
    author = forms.CharField(label="저자", validators=[min_length_3_validator])
    publisher = forms.CharField(label="출판사", required = False)

    def save(self, commit=True):
        book = Book(**self.cleaned_data)
        if commit:
            book.save()
        return book
```

```
# models.py
```

```
from django.db import models
```

```
from django.urls import reverse
```

```
class Book(models.Model):
```

```
    title = models.CharField(max_length=50)
```

```
    author = models.CharField(max_length=50)
```

```
    publisher = models.CharField(max_length=50)
```

```
    publication_date = models.DateTimeField(auto_now_add=True)
```

```
    def get_absolute_url(self):
```

```
        return reverse('book:list')
```

```
# views.py(1)
```

```
from django.shortcuts import render, redirect
from django.views.generic import CreateView, ListView
from .models import Book
from .forms import BookForm
```

```
# book_new = CreateView.as_view(model=Book, fields='__all__')
```

```
def book_new(request):
    if request.method == 'POST':
        form = BookForm(request.POST, request.FILES)
        if form.is_valid():
            print(form.cleaned_data)

            # 방법1
            # book = Book()
            # book.title = form.cleaned_data['title']
            # book.author = form.cleaned_data['author']
            # book.publisher = form.cleaned_data['publisher']
            # book.save()
```

```
# views.py(2)

# 방법2
# book = Book(title=form.cleaned_data['title'], author=form.cleaned_data['author'],
               publisher=form.cleaned_data['publisher'])
# book.save()

# 방법3
book = form.save()
return redirect(book)

else:
    form = BookForm()

return render(request, 'book/book_form2.html',{'form':form})

book_list = ListView.as_view(model=Book)
```

```
# book_form2.html
```

```
{% extends "layout.html" %}
```

```
{% block title %}책저장{% endblock %}
```

```
{% block content %}
```

```
    <form action="" method="post">
```

```
        {% csrf_token %}
```

```
        <table>
```

```
            {{form.as_table}}
```

```
            <tr><td colspan="2">
```

```
                <input type="submit" value="저장">
```

```
                <input type="button" value="목록"
```

```
                onclick="location='{% url 'book:list'%}'">
```

```
            </td></tr>
```

```
        </table>
```

```
    </form>
```

```
{% endblock %}
```



```
# book_list.html
```

```
{% if book_list %}
    <ul>
        {% for book in book_list%}
            <li>{{book.title}}/{{book.author}} /{{book.publisher}}</li>
        {% endfor %}
    </ul>
{% else %}
    <p>목록이 없습니다.</p>
{% endif %}
```

```
# urls.py
```

```
from django.urls import path
from . import views

app_name='book'
urlpatterns =[
    path('', views.book_new, name="new"),
    path('list/', views.book_list, name='list')
]
```



5. Django Model Form



ModelForm

5. Django Model Form

- ModelForm은 forms.Form을 상속받음
- ModelForm은 모델을 기초로 폼을 생성함

```
class BookModelForm(forms.ModelForm):  
    class Meta:  
        model = Book  
        fields = '__all__'  
        # fields = ['title', 'author', 'publisher']
```

- 내부적으로 모델 인스턴스 유지
- 유효성 검증을 통과한 값들을 모델 인스턴스로 저장 지원
(create or Update)

Form vs. ModelForm

5. Django Model Form

```
class BookForm(forms.Form):
    title = forms.CharField()
    author = forms.CharField()
    publisher = forms.CharField()

    def save(self, commit=True):
        book = Book(**self.cleaned_data)
        if commit:
            book.save()
        return book
```



```
class BookModelForm(forms.ModelForm):
    class Meta:
        model = Book
        fields = ['title', 'author', 'publisher']
```

ModelForm 속성

5. Django Model Form

- **fields**
 - 리턴하는 ModelForm에 포함될 필드 지정. 필수 속성
- **exclude**
 - 리턴하는 ModelForm에 제외될 필드 지정
- **widgets**
 - 모델 필드와 위젯을 매핑한 사전
- **formfield_callback**
 - 모델의 필드를 받아서 폼 필드를 리턴하는 콜백 함수 지정
- **localized_fields**
 - 로컬 지역값이 필요한 필드를 리스트로 지정
- **labels**
 - 모델 필드와 레이블을 매핑한 사전
- **help_texts**
 - 모델 필드와 설명 문구를 매핑한 사전
- **error_messages**
 - 모델 필드와 에러 메시지를 매핑한 사전

<https://docs.djangoproject.com/en/3.2/ref/models/fields/#common-model-field-options> 의 option은 하나는 DB에 반영되는 option, 하나는 form에 반영되는 option

Model 필드와 Form 필드 매핑

5. Django Model Form

Model field	Form field
AutoField	form에 표시되지 않음
BigAutoField	form에 표시되지 않음
BigIntegerField	IntegerField(-9223372036854775808 ~ 9223372036854775807)
BinaryField	CharField, 편집 가능상태가 아닌 경우 form에 표시되지 않음
BooleanField	BooleanField, 또는 null=True 이면 NullBooleanField
CharField	CharField
DateTimeField	DateTimeField
DecimalField	DecimalField
EmailField	EmailField
FileField	FileField
FilePathField	FilePathField
FloatField	FloatField

Model 필드와 Form 필드 매핑

5. Django Model Form

Model field	Form field
ForeignKey	ModelChoiceField
ImageField	ImageField
IntegerField	IntegerField
IPAddressField	IPAddressField
GenericIPAddressField	GenericIPAddressField
ManyToManyField	ModelMultipleChoiceField
NullBooleanField	NullBooleanField
PositiveIntegerField	IntegerField
PositiveSmallIntegerField	IntegerField
SlugField	SlugField
SmallIntegerField	IntegerField
TextField	CharField
TimeField	TimeField
URLField	URLField

ModelForm 유효성 검증

5. Django Model Form

book/models.py

```
from django.db import models
from django.urls import reverse
from django import forms
```

```
def min_length_3_validator(value):
    if len(value) < 3:
        raise forms.ValidationError('3글자 이상 입력해주세요')
```

```
class Book(models.Model):
    title = models.CharField(max_length=50)
    author = models.CharField(max_length=50, validators=[min_length_3_validator])
    publisher = models.CharField(max_length=50)
    publication_date = models.DateTimeField(auto_now_add=True)

    def get_absolute_url(self):
        return reverse('book:list')
```


commit 지연

5. Django Model Form

```
# book/models.py
```

```
class Book(models.Model):  
    title = models.CharField(max_length=50)  
    author = models.CharField(max_length=50, validators= ...  
    publisher = models.CharField(max_length=50)  
    publication_date = models.DateTimeField(auto_now_add=True)  
    ip = models.CharField(max_length=15)  
  
    def get_absolute_url(self):  
        return reverse('book:list')
```

```
D:\src\myproject>python manage.py makemigrations book
```

```
D:\src\myproject>python manage.py migrate book
```

commit 지연

5. Django Model Form

```
# book/forms.py
class BookModelForm(forms.ModelForm):
    class Meta:
        model = Book
        fields = ['title', 'author', 'publisher', 'ip']
```

Title:

Author:

Publisher:

Ip:

commit 지연

5. Django Model Form

```
# book/views.py
```

```
def book_new(request):  
    if request.method == 'POST':  
        form = BookModelForm(request.POST, request.FILES)  
        if form.is_valid():  
            book = form.save(commit=False)  
            book.ip = request.META['REMOTE_ADDR']  
            book.save()  
            return redirect(book)  
        else:  
            form = BookModelForm()  
            return render(request, 'book/book_form2.html', {'form': form})
```

HttpRequest 객체
(<https://docs.djangoproject.com/en/3.2/ref/request-response/#httprequest-objects>)

commit 지연

5. Django Model Form

```
class BookForm(forms.Form):
    title = forms.CharField(label="제목")
    author = forms.CharField(label="저자")
    publisher = forms.CharField(label="출판사", required = False)

    def save(self, commit=True):
        book = Book(**self.cleaned_data)
        if commit:
            book.save()
        return book
```

ModelForm 수정

5. Django Model Form

```
# book/urls.py

from django.urls import path
from . import views

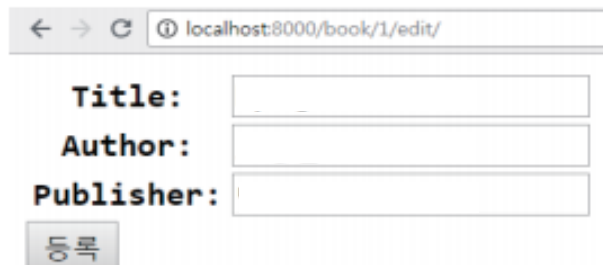
app_name='book'
urlpatterns =[
    path('', views.book_new, name="new"),
    path('list/', views.book_list, name='list'),
    path('<id>/edit/', views.book_edit, name='edit'),
]
```

ModelForm 수정

5. Django Model Form

```
# book/views.py
from django.shortcuts import get_object_or_404

def book_edit(request, id):
    book = get_object_or_404(Book, id=id)
    if request.method == 'POST':
        form = BookModelForm(request.POST, request.FILES, instance=book)
        if form.is_valid():
            book = form.save(commit=False)
            book.ip = request.META['REMOTE_ADDR']
            book.save()
            return redirect(book)
    else:
        form = BookModelForm(instance=book)
    return render(request, 'book/book_form2.html', {'form': form})
```



← → ↻ localhost:8000/book/1/edit/

Title:

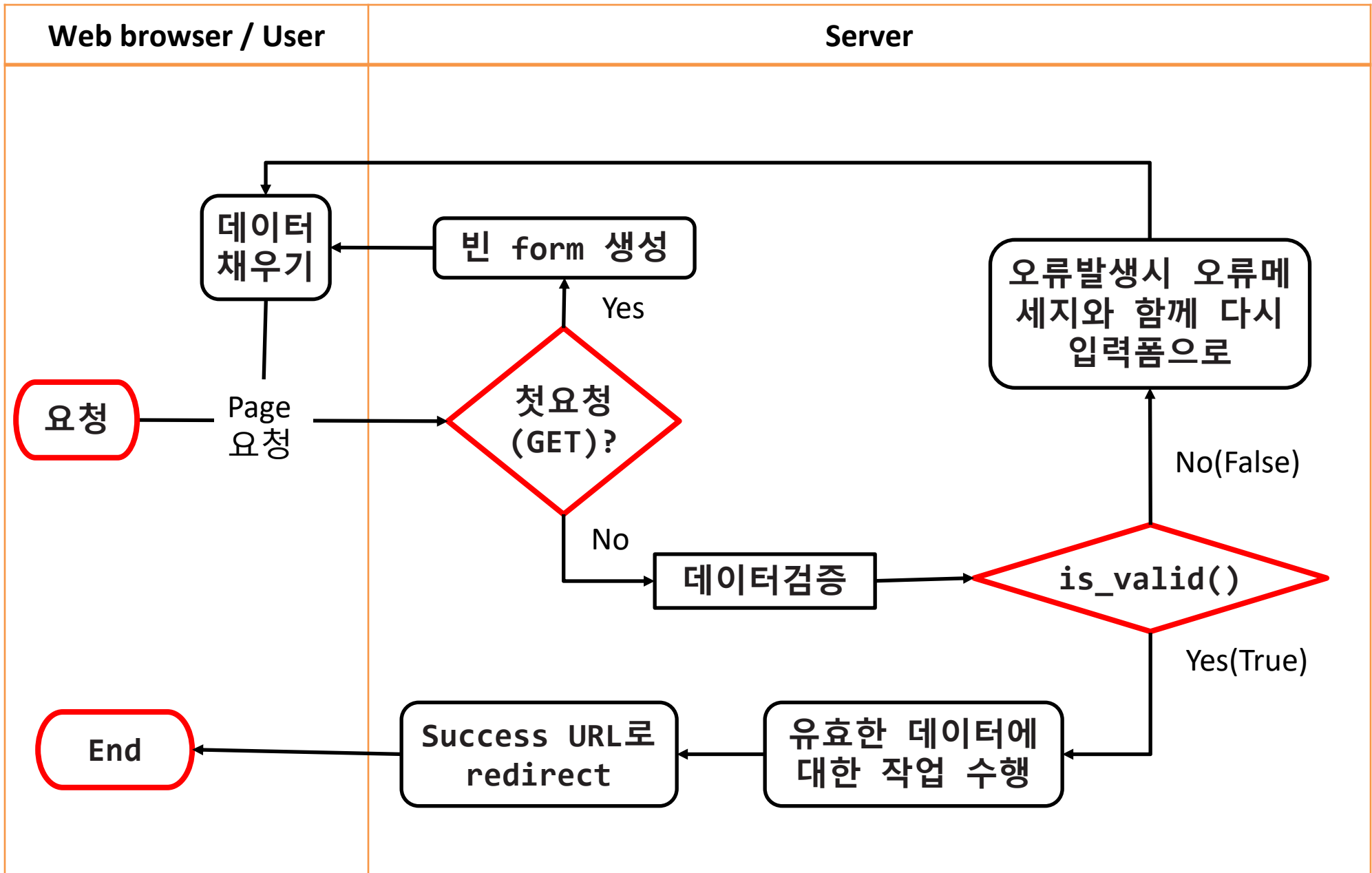
Author:

Publisher:



6. Form Validation





유효성 검사

6. Form Validation

```
def book_new(request):  
    if request.method == 'POST':  
        form = BookModelForm(request.POST, request.FILES)  
        if form.is_valid():  
            form.save()  
    else:  
        form = BookModelForm()
```

form.is_valid() 호출

1. model이나 form의 필드에 정의된 validators 호출 : return값 없음.
2. form내의 clean_필드명() 호출 : 하나의 필드에 대해서만 유효성 검증. Return값은 해당 필드값
3. Form내의 clean() 호출 : 여러 개의 필드값에 대해 복합적으로 유효성 검증. return값은 여러 개의 필드값

유효성 검사

6. Form Validation

- **validator 함수를 통한 검사**
 - 유효성 검사 실패시 `ValidationError` 예외 발생
 - 리턴값 없음
- **Form의 `clean` 멤버함수 통한 검사 및 값 변경**
 - 유효성 검사 실패시 `ValidationError` 예외 발생
 - 리턴값을 통해 값 변경

validator

6. Form Validation

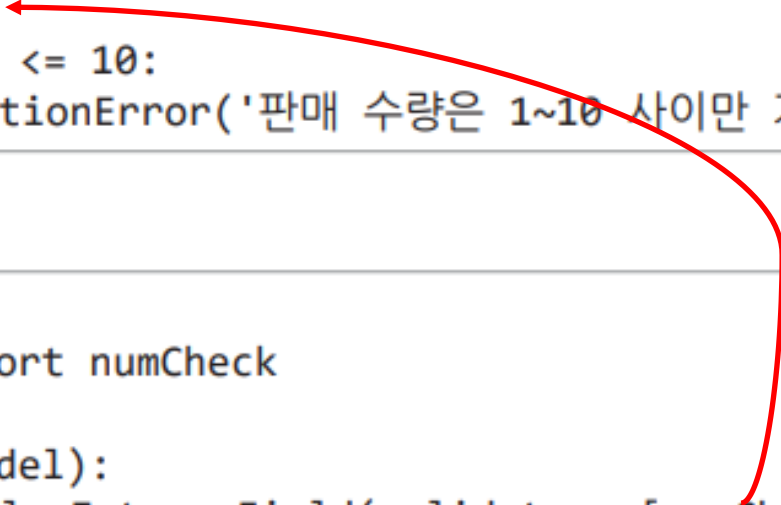
- 유효성 검사를 수행할 값을 인자로 받음
- 유효성 검사 실패시 ValidationError 발생
- ValidationError 발생시 해당 필드 오류로 분류
- 리턴값 무시
- 함수 validator는 snake_case, 클래스 validator는 CamelCase 형식으로 지정

Model Field Validators

6. Form Validation

```
# book/validator.py
import re
from django import forms
from django.core.exceptions import ValidationError

def numCheck(value):
    if not 1<= value <= 10:
        raise ValidationError('판매 수량은 1~10 사이만 가능합니다');
```



```
# book/models.py
from .validators import numCheck

class Book(models.Model):
    sales = models.IntegerField(validators=[numCheck] )
```

Model 필드의 기본 validators

6. Form Validation

- `models.EmailField` : `CharField`상속 + `validate_email`
- `models.URLField` : `CharField`상속 + `URLValidator()`
- `models.GenericIPAddressField` : `ip_address_validators`
- `models.SlugField` : `CharField`상속 + `validate_slug` 혹은 `validate_unicode_slug`

Form의 clean 함수

6. Form Validation

- **필드별 에러 기록 또는**
 - 유효성 검사 실패시 `ValidationError` 통해 오류 기록
 - `add_error(필드명, 오류내용)` 통해 오류 기록
- **값 변경**
 - 리턴값을 통해 값 변경 지원
- **`clean_필드명()` : 필드별 검사 및 변경**
 - `ValidationError` 발생시 해당 필드 Error로 분류
- **`clean()` : 다수 필드 검사 및 변경**
 - `ValidationError` 발생시 `non_field_errors`로 분류
 - `add_error` 함수를 통해 필드별 Error 기록 가능

Validators vs. Clean

6. Form Validation

- Model에 validators는 정의하는 방법 추천
- ModelForm을 통해 Model의 validators 사용
- clean을 사용해야 하는 경우
 - 특정 Form에서 1회성 유효성 검사시
 - 다수 필드를 묶어 유효성 검사시
 - 필드 값을 변경할 필요가 있는 경우

clean 함수 예제 코드

6. Form Validation

```
class BookModelForm(forms.ModelForm):
    class Meta:
        model = Book
        fields = ['title', 'author', 'publisher', 'sales']

    def clean_author(self):
        author = self.cleaned_data.get('author', '').strip()
        if author:
            if len(author) < 3:
                raise ValidationError('최소 3글자 이상 입력하세요')
        return author

    def clean(self):
        cleaned_data = super().clean()
        if self.check_exist(cleaned_data.get('title'), cleaned_data.get('author')):
            raise ValidationError('이미 등록된 책입니다.')
        return cleaned_data

    def check_exist(self, title, author):
        return Book.objects.filter(title=title, author=author).exists()
```


clean 함수 예제 코드 변경

6. Form Validation

```
from django.core.validators import MinLengthValidator

class Book(models.Model):
    title = models.CharField(max_length=50)
    author = models.CharField(max_length=50, validators=[MinLengthValidator(3)])
    publisher = models.CharField(max_length=50)
    publication_date = models.DateTimeField(auto_now_add=True)
    ip = models.CharField(max_length=15)
    sales = models.IntegerField(validators=[MinValueValidator(1)] )

    class Meta:
        unique_together= (('title', 'author'),)

    def get_absolute_url(self):
        return reverse('book:list')
```