



통합 구현(Spring, Django)

# 유저 인증과 Session



한국기술교육대학교  
온라인평생교육원

## 학습내용

- 유저 인증
- 회원 가입
- Session

## 학습목표

- Django에서 로그인을 구현하는 방법을 설명할 수 있다.
- Django에서 회원가입을 구현하는 방법을 설명할 수 있다.
- Django에서 Session을 활용하는 방법을 파악하여 적용할 수 있다.

# 유저 인증

## 1 기본 인증

### 1 인증 앱

- 1 Django는 기본적으로 Django.contrib.auth 앱을 통해 인증 기능을 제공함
- 2 인증 관련 URL과 뷰가 이미 존재함
- 3 User(사용자) 테이블을 기본으로 제공하기 때문에 별도로 테이블을 작성할 필요가 없음

### 2 인증 관련 URL

URL		템플릿
/accounts/login	login()	registration/login.html
/accounts/logout	logout()	registration/logged_out.html
/accounts/password_change	password_change()	registration/password_change_form.html
/accounts/password_change/done	password_change_done()	registration/password_change_done.html
/accounts/password_reset	password_reset()	registration/password_reset_form.html
/accounts/password_reset/done	password_reset_done()	registration/password_reset_done.html

Django 기본 제공 URL과 뷰 매핑

# 유저 인증

## 1 기본 인증

### 3 User 테이블

필드	타입	설명
id	Integer	기본 키
password	CharField(128)	비밀번호
username	CharField(30)	로그인 이름
first_name	CharField(30)	사용자 이름
last_name	CharField(30)	사용자 성
email	CharField(254)	이메일
is_superuser	BooleanField	슈퍼유저(관리자) 여부
is_staff	BooleanField	스태프 여부
is_active	BooleanField	계정 활성화 여부
date_joined	DateTimeField	계정 생성 시간
last_login	DateTimeField	마지막 로그인 시간

Django 기본 제공 User 테이블

## 유저 인증

### 2 로그인 구현

#### 1 기본 URL 설정

1 다음과 같은 URL을 settings.py에 설정

LOGIN\_URL

로그인 페이지 화면  
(기본값: '/accounts/login/')

LOGOUT\_URL

로그아웃을 위한 URL  
(기본값: '/accounts/logout/')

LOGIN\_REDIRECT\_URL

로그인 성공 후 이동할 URL  
(기본값: '/accounts/profile/')

2 기본 URL과 뷰는 수정할 수 있으며 기본값을 그대로 사용할 경우는 별도로 설정하지 않아도 좋음

```
LOGIN_URL = '/accounts/login/'
LOGOUT_URL = '/accounts/logout/'
LOGIN_REDIRECT_URL = '/'
```

## 유저 인증

### 2 로그인 구현

#### 2 URLConf 설정

/accounts 주소에 대해서 Django.contrib.auth.urls 뷰 추가

```
from django.conf.urls import include, url
from django.contrib import admin
from polls import views

urlpatterns = [
    url(r'^accounts/', include('django.contrib.auth.urls')),
    url(r'^polls/', include('polls.urls', namespace="polls")),
    url(r'^parent/', views.TemplateParentView.as_view(), name='parent'),
    url(r'^child/', views.TemplateChildView.as_view(), name='child'),
    url(r'^admin/', include(admin.site.urls)),
]
```

## 유저 인증

### 2 로그인 구현

#### 3 템플릿 설정

1

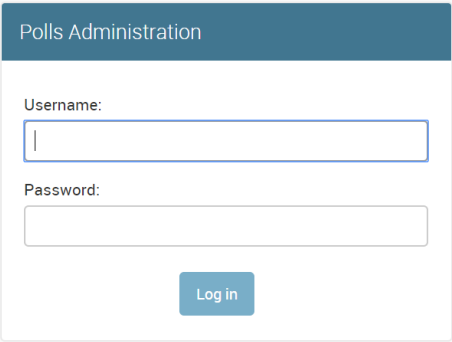
Django에서 기본 제공하는 관리자용 로그인 화면을 재사용하면 손쉽게 로그인 화면을 구현할 수 있음

2

프로젝트폴더/templates/registration 폴더 밑에 login.html과 base\_site.html을 추가

3

/accounts/login 주소에 접속하면 다음과 같은 로그인 화면을 볼 수 있으며, 아직은 관리자 외의 계정이 없으므로 관리자 계정으로만 로그인 가능



로그인 화면

# 회원 가입

## 1 뷰 구현

### 회원 가입을 위한 뷰 생성

1

Django에서는 회원 관련 기능이 대부분 구현되어있기 때문에 뷰 구현도 간소화되어 있음

2

mysite/mysite 폴더에 views.py 파일을 생성 후 GenericView 중 CreateView를 상속하여 회원 가입 뷰를 다음과 같이 구현

3

CreateView에서 요구하는 form\_class에 UserCreationForm이라는 작성된 폼 클래스를 등록하면 템플릿에서 자동으로 회원가입 폼 생성

4

success\_url에는 회원가입 성공 후 이동할 URL을 기입하는데 이 때, reverse\_lazy함수에 URL Name을 입력함

```
from django.views.generic.edit import CreateView
from django.contrib.auth.forms import UserCreationForm
```

```
class RegistrationView(CreateView):
    template_name = 'registration/registration.html'
    form_class = UserCreationForm
    success_url = reverse_lazy('polls:index')
```



# 회원 가입

## 1 뷰 구현

### 회원 가입을 위한 URLConf 수정

accounts/registration URL에 앞서 작성한 뷰를 등록함

```
from django.conf.urls import include, url
from django.contrib import admin
from mysite.views import UserCreateView
from polls import views

urlpatterns = [
    url(r'^accounts/', include('django.contrib.auth.urls')),
    url(r'^accounts/registration', UserCreateView.as_view(),
name='registration'),
    url(r'^polls/', include('polls.urls', namespace="polls")),
    url(r'^parent/', views.TemplateParentView.as_view(), name='parent'),
    url(r'^child/', views.TemplateChildView.as_view(), name='child'),
    url(r'^admin/', include(admin.site.urls)),
]
```

# 회원 가입

## 2 템플릿 구현(registration.html)

1 기존의 Admin Site 템플릿을 재사용한 템플릿으로 가장 핵심은 Form 구성 부분

```
{% extends "admin/base_site.html" %}
{% load i18n static %}

{% block extrastyle %}{{ block.super }}<link rel="stylesheet" type="text/css"
href="{% static "admin/css/login.css" %}" />
{{ form.media }}
{% endblock %}

{% block bodyclass %}{{ block.super }} login{% endblock %}

{% block usertools %}{% endblock %}

{% block nav-global %}{% endblock %}

{% block content_title %}{% endblock %}

{% block breadcrumbs %}{% endblock %}
```

2 {{form.as\_p}}와 같이 입력하면 기본적으로 제공하는 Form이 표현됨

```
{% block content %}
<div id="content-main">
<form action="{% url 'registration' %}" method="post" id="login-form">
  {% csrf_token %}
  {{ form.as_p }}
  <input type="submit" value="Register"/>
</form>
</div>
{% endblock %}
```

## 회원 가입

### 2 템플릿 구현 (registration.html)

#### 3 accounts/registration 접속 화면

Polls Administration

Username:  Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Password: 

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:  Enter the same password as before, for verification.

Register

accounts/registration 접속 화면

# Session

## 1 Session 변수 사용

### 1 views.py에서의 사용

1 Session 변수는 어떤 페이지를 이동해도 접근할 수 있다는 장점이 있음

2 Django에서는 기본 request 객체를 통해 Session 변수를 저장할 수 있음

3 다음과 같이 views.py에 SessionTestView를 생성함

Session 변수  
저장 방법

```
request.session['변수이름'] = 값
```

```
from django.views.generic.edit import CreateView
from django.views.generic.edit import CreateView
from django.contrib.auth.forms import UserCreationForm
from django.core.urlresolvers import reverse_lazy
from django.shortcuts import render
from django.views.generic import View
```

```
class RegistrationView(CreateView):
    ...생략...
```

```
class SessionTestView(View):
    template_name = 'session.html'
    def get(self, request):
        request.session['session_variable'] = 'test'
        return render(request, self.template_name)
```

# Session

## 1 Session 변수 사용

### 2 URLConf 설정

앞서 등록한 모델에 'Add' 버튼을 클릭하면 모델 구성에 따른 데이터를 입력할 수 있음

- session/ URL에 앞서 작성한 SessionTestView를 등록함

```
from django.conf.urls import include, url
from django.contrib import admin
from mysite.views import RegistrationView
from mysite.views import SessionTestView
from polls import views

urlpatterns = [
    url(r'^accounts/', include('django.contrib.auth.urls')),
    url(r'^accounts/registration', RegistrationView.as_view(),
name='registration'),
    url(r'^session/', SessionTestView.as_view(), name="session"),
    url(r'^polls/', include('polls.urls', namespace="polls")),
    url(r'^parent/', views.TemplateParentView.as_view(), name='parent'),
    url(r'^child/', views.TemplateChildView.as_view(), name='child'),
    url(r'^admin/', include(admin.site.urls)),
]
```

# Session

## 1 Session 변수 사용

### 3 템플릿 설정

1 프로젝트명/templates/session.html을 다음과 같이 작성함

2 Session에 저장된 변수를 이용하는 방법은 다음과 같음

- `{{ request.session.Session 변수이름 }}`

3 Session 변수는 여러 페이지를 이동해도 유지되기 때문에 다른 페이지에서도 사용할 수 있음

`{{ request.session.session_variable }}`

## 2 Session 관리

### 1 Session 변수 삭제

Session 변수는 브라우저를 닫지 않는 한 계속 유지되므로 Scope 관리가 복잡해질 수 있음

적절한 때에 View에서 Session 변수를 삭제할 수 있음

- `del request.session['Session 변수명']`

### 2 Session 삭제

Django에서 로그아웃 처리는 현재 Session을 삭제하고 새로운 세션을 시작하는 것과 같음

로그아웃을 직접 구현하고자 Session 자체를 삭제하고 새로운 Session을 시작할 때 다음과 같이 View에서 적용할 수 있음

- `request.session.flush()`

### 1. 유저 인증

- 기본 인증
  - Django는 기본적으로 Django.contrib.auth 앱을 통해 인증 기능 제공
  - 인증 관련 URL과 뷰가 이미 존재함
  - User(사용자) 테이블을 기본으로 제공하기 때문에 별도로 테이블을 작성할 필요가 없음
- 로그인 구현
  - 기본 URL 설정
  - URLConf 설정
  - 템플릿 설정

### 2. 회원 가입

- 회원 가입을 위한 뷰 생성
  - Django에서는 회원 관련 기능이 대부분 구현되어있기 때문에 뷰 구현도 간소화되어 있음
  - mysite/mysite 폴더에 views.py 파일을 생성 후 GenericView 중 CreateView를 상속하여 회원 가입 뷰를 구현
  - CreateView에서 요구하는 form\_class에 UserCreationForm이라는 미리 작성된 폼 클래스를 등록하면 추후 템플릿에서 자동으로 회원가입 폼이 생성됨
  - success\_url에는 회원가입 성공 후 이동할 URL을 기입
- 회원 가입을 위한 URLConf 수정
- 템플릿 구현
  - 기존의 Admin site 템플릿을 재사용한 템플릿으로 가장 핵심은 Form 구성 부분임
  - {{form.as\_p}}와 같이 입력하면 기본적으로 제공하는 Form이 표현됨



### 3. Session

- views.py에서의 사용
  - Session 변수는 어떤 페이지를 이동해도 접근할 수 있다는 장점이 있음
  - Django에서는 기본 request 객체를 통해 Session 변수를 저장할 수 있음
  - views.py에 SessionTestView를 생성
- Session 변수 삭제
  - Session 변수는 브라우저를 닫지 않는 한 계속 유지되므로 Scope 관리가 복잡해질 수 있음
  - 적절한 때에 View에서 Session 변수를 삭제할 수 있음
- Session 삭제
  - Django에서 로그아웃 처리는 현재 Session을 삭제하고 새로운 Session을 시작하는 것과 같음
  - 로그아웃을 직접 구현하고자 Session 자체를 삭제하고 새로운 Session을 시작할 때 View에서 적용할 수 있음