

Django 6: Authentication

IN608: Intermediate Application Development Concepts

Kaiako: Tom Clark & Grayson Orr

Last Session's Content

- Automation testing
 - Model testing
 - View testing
- LiveServerTestCase

Today's Content

- Authentication
 - AbstractUser
 - Auth views
 - Django crispy forms
 - Session-based authentication

Authentication

Authentication

- Django's `User` model may not always be appropriate
- A site's login may require an email address instead of a username
- Override the default `User` model by providing a value for `AUTH_USER_MODEL` in `mvt/settings.py`
- Resource: <https://docs.djangoproject.com/en/3.0/topics/auth/customizing>

```
AUTH_USER_MODEL = 'polls.User'
```

Authentication

- polls/models.py

```
class User(AbstractUser):  
    username = models.CharField(max_length=25, unique=True, blank=False)  
    email = models.EmailField(unique=True, blank=False)  
    first_name = models.CharField(max_length=200, blank=False)  
    last_name = models.CharField(max_length=200, blank=False)
```

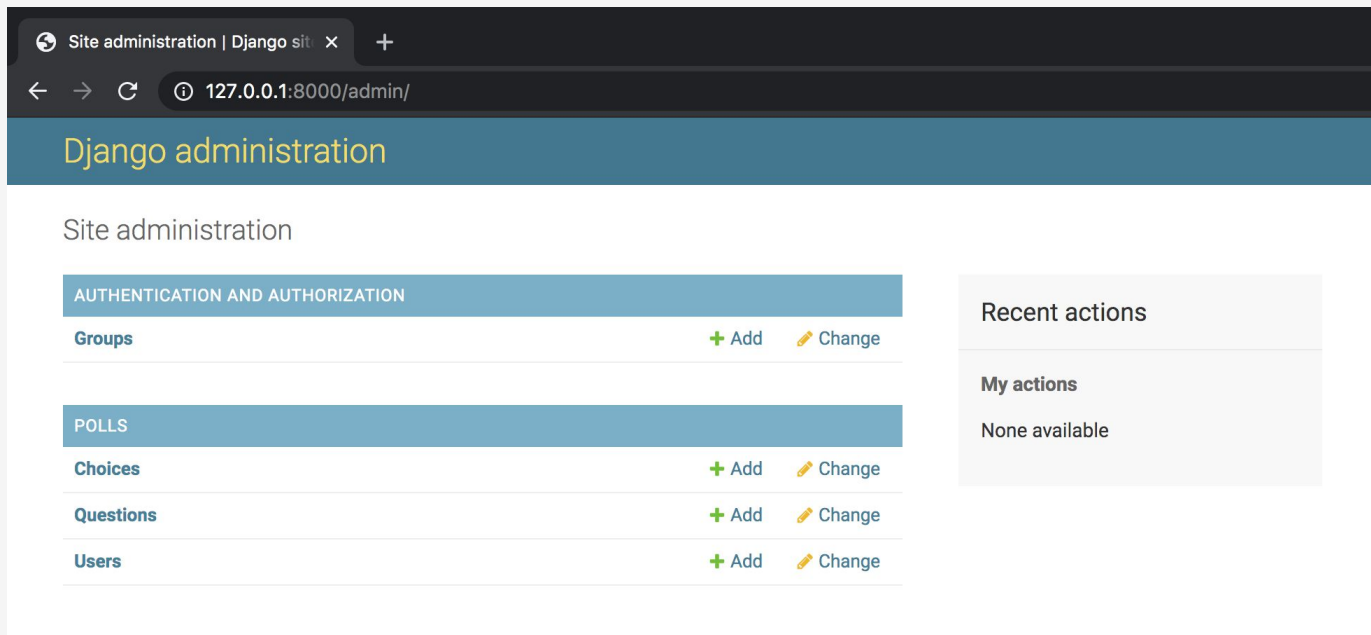
Authentication

- polls/admin.py

```
from django.contrib.auth.admin import UserAdmin
from .models import User, Question, Choice

admin.site.register(User, auth.admin.UserAdmin)
admin.site.register(Question)
admin.site.register(Choice)
```

Authentication



Authentication

- `polls/urls.py`
- `auth_views.LoginView`
 - If login is successful, the view redirects to the URL specified in `mvt/settings.py`, i.e., `LOGIN_REDIRECT_URL`
 - If login is not successful, it displays the login form
- `auth_views.LogoutView`
 - The view redirects to the URL specified in `mvt/settings.py`, i.e., `LOGOUT_REDIRECT_URL`
- Resources:
 - <https://docs.djangoproject.com/en/3.0/topics/auth/default/#django.contrib.auth.views.LoginView>
 - <https://docs.djangoproject.com/en/3.0/topics/auth/default/#django.contrib.auth.views.LogoutView>

```
from django.contrib.auth import views as auth_views
from django.urls import path
from . import views

app_name = 'polls'

urlpatterns = [
    path('accounts/signup/', views.Signup.as_view(), name='signup'),
    path('accounts/login', auth_views.LoginView.as_view(), name='login'),
    path('accounts/logout', auth_views.LogoutView.as_view(), name='logout'),
    path('', views.IndexView.as_view(), name='index'), # /polls/
    path('<int:pk>', views.DetailView.as_view(), name='detail'), # /polls/2/
    path('<int:pk>/results/', views.ResultsView.as_view(), name='results'), # /polls/2/results/
    path('<int:question_id>/vote/', views.vote, name='vote'), # /polls/2/vote/
]
```

Authentication

- `mvt/settings.py`
- **Resources:**
 - https://docs.djangoproject.com/en/3.0/ref/settings/#std:setting-LOGIN_REDIRECT_URL
 - https://docs.djangoproject.com/en/3.0/ref/settings/#std:setting-LOGOUT_REDIRECT_URL

```
LOGIN_REDIRECT_URL = '/polls/'  
LOGOUT_REDIRECT_URL = '/accounts/login/'
```

Authentication

- In the `polls/` directory, create a file called `forms.py`
- App structure:

```
polls/  
  __init__.py  
  admin.py  
  apps.py  
  forms.py  
  migrations/  
    __init__.py  
  models.py  
  static/  
    polls/  
      style.css  
  templates/  
    polls/  
      base.html  
      detail.html  
      index.html  
      results.html  
  tests.py  
  urls.py  
  views.py
```

Authentication

- polls/forms.py directory, create a file called forms.py
- UserCreationForm
 - Three fields (username from the `User` model, password1 & password 2)
 - Validates the password using `validate_password()`
 - Sets the user's password using `validate_password()`

```
from django.contrib.auth import forms
from .models import User

class SignupForm(forms.UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'email', 'first_name', 'last_name']
```

Authentication

- polls/views.py
- FormView
 - A view that display a form
 - On error, displays the form with validation errors
 - On success, redirects to a URL
- `from django.contrib.auth import login`
- `from .forms import SignupForm`
- `from .models import User`
- Resource: <https://docs.djangoproject.com/en/3.0/ref/class-based-views/generic-editing>

```
class SignupView(generic.FormView):
    model = User
    form_class = SignupForm
    template_name = 'registration/signup.html'

    def form_valid(self, form):
        user = form.save()
        login(self.request, user)
        return HttpResponseRedirect(reverse('polls:index'))
```

Authentication

- `polls/views.py`
- `@method_decorator(decorators.login_required)`
- `@method_decorator(decorators.login_required, name='dispatch')`
 - If the user is not logged in, redirect to `LOGIN_REDIRECT_URL` + the current absolute path in the query string, i.e., </accounts/login/?next=/polls/>
 - If the user is logged in, display the view
- `from django.contrib.auth import decorators`
- `from django.utils.decorators import method_decorator`
- Resources:
 - <https://docs.djangoproject.com/en/3.0/topics/auth/default/#the-login-required-decorator>
 - <https://docs.djangoproject.com/en/3.0/topics/class-based-views/intro/#decorating-the-class>

```
@method_decorator(decorators.login_required, name='dispatch')
class IndexView(generic.ListView):
    ...
```

Authentication

- In the `polls/templates/` directory, create a directory called `registration/`
- In the `registration/` directory, create two `.html` files called `login` & `signup`
- App structure:

```
polls/  
  __init__.py  
  admin.py  
  apps.py  
  forms.py  
  migrations/  
    __init__.py  
  models.py  
  static/  
    polls/  
      style.css  
  templates/  
    polls/  
      base.html  
      detail.html  
      index.html  
      results.html  
      registration/  
        login.html  
        signup.html  
  tests.py  
  urls.py  
  views.py
```

Authentication

- Install Django crispy forms

```
# Windows
...\> pipenv install django-crispy-forms

# Linux or macOS
$ pipenv install django-crispy-forms
```


Authentication

- View Pipfile & Pipfile.lock

```
[[source]]
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true

[dev-packages]

[packages]
django = "*"
selenium = "*"
django-crispy-forms = "*"

[requires]
python_version = "3.7"
```

Authentication

- `mvt/settings.py`
- Resource:
 - <https://pypi.org/project/django-crispy-forms>
 - <https://django-crispy-forms.readthedocs.io/en/latest>

```
INSTALLED_APPS = [  
    'polls.apps.PollsConfig',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'crispy_forms',  
]
```

Authentication

- polls/templates/registration/signup.html

```
{% extends "polls/base.html" %}

{% block title %}Polls | Signup{% endblock %}

{% load crispy_forms_tags %}

{% block content %}
<form method="POST">
  {% csrf_token %}
  {{ form|crispy }}
  <input type="submit" value="Signup">
</form>
<br>
<a href="{% url 'polls:login' %}">Login</a>
{% endblock %}
```

Authentication

Polls | Signup

127.0.0.1:8000/polls/accounts/signup/

Username*

First name*

Last name*

Email*

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.

Signup

[Login](#)

Polls | Signup

127.0.0.1:8000/polls/accounts/signup/

Username*

johndoe123

First name*

John

Last name*

Doe

Email*

jdoe@example.com

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*

The two password fields didn't match.

Enter the same password as before, for verification.

Signup

[Login](#)

Authentication

- polls/templates/registration/login.html

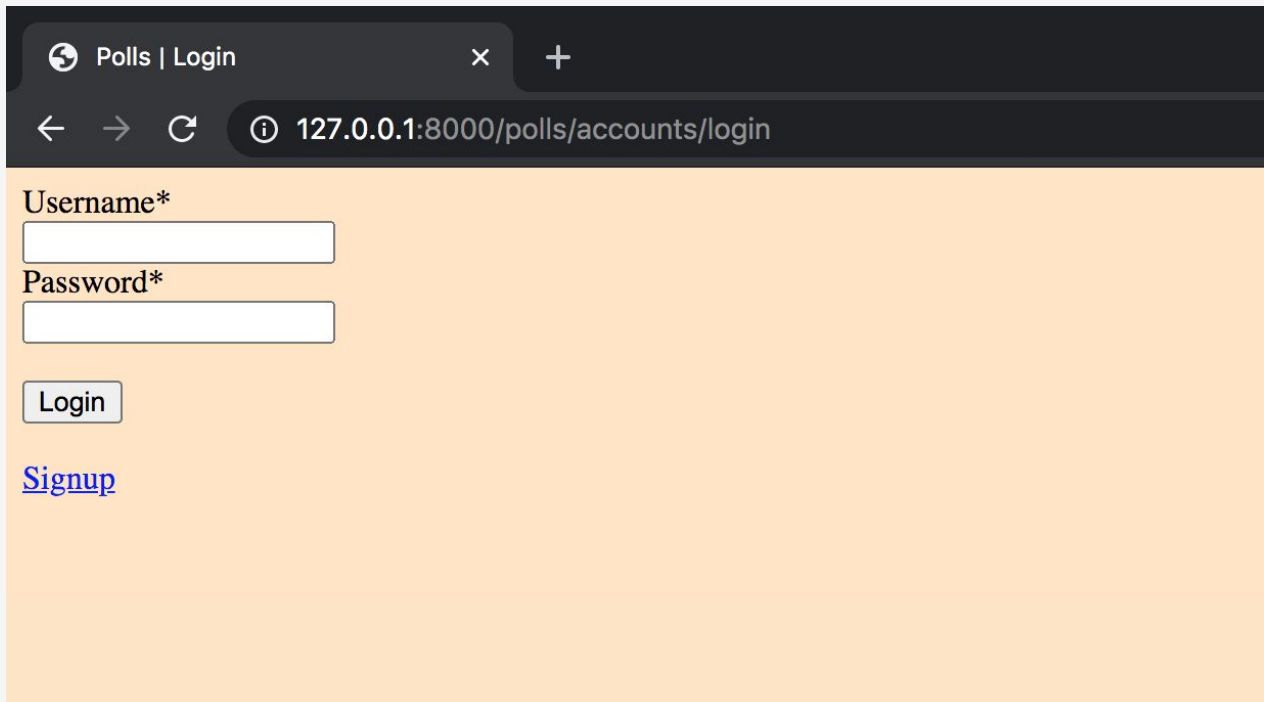
```
{% extends "polls/base.html" %}

{% block title %}Polls | Login{% endblock %}

{% load crispy_forms_tags %}

{% block content %}
<form method="POST">
  {% csrf_token %}
  {{ form.username|as_crispy_field }} {{ form.password|as_crispy_field }}<br>
  <input type="submit" value="Login">
</form>
<br>
<a href="{% url 'polls:signup' %}">Signup</a>
{% endblock %}
```

Authentication



A screenshot of a web browser window with a dark theme. The browser's address bar shows the URL `127.0.0.1:8000/polls/accounts/login`. The page has a light orange background and contains a login form with the following elements:

- A label `Username*` followed by a text input field.
- A label `Password*` followed by a text input field.
- A `Login` button.
- A [Signup](#) link.

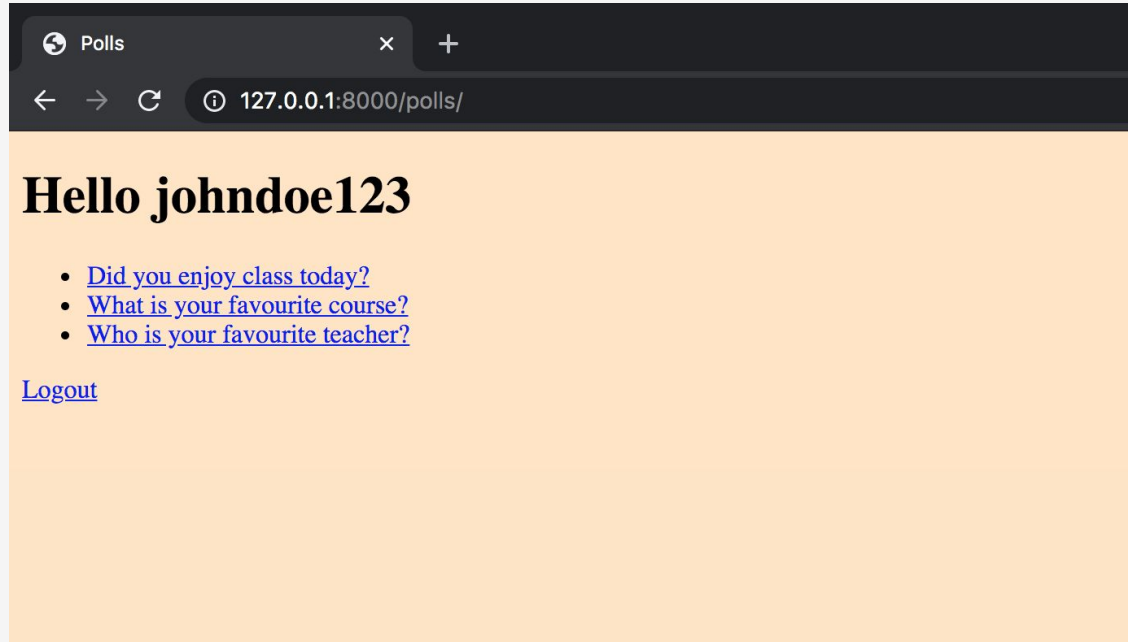
Authentication

- polls/templates/registration/index.html

```
{% extends "polls/base.html" %}

{% block content %}
<h1>Hello {{ user.username }}</h1>
{% if latest_question_list %}
  <ul>
    {% for question in latest_question_list %}
      <li><a href="{% url 'polls:detail' question.id %}">{{ question }}</a></li>
    {% endfor %}
  </ul>
{% else %}
  <p>No polls are available.</p>
{% endif %}
<a href="{% url 'polls:logout' %}">Logout</a>
{% endblock %}
```

Authentication



Authentication

- Session-based authentication
 - A user's logged in state is saved in the server's memory
 - A session is securely created by the server when a user signs in
 - A session ID is stored in a cookie in the user's browser
 - While the user is logged in, the cookie is sent with every request
 - The server looks at the session cookie & reads the session ID
 - If it matches the data stored in the server's memory, it sends a response back to the browser
- Alternatives - JSON Web Tokens (token-based authentication)
 - Resource: <https://jwt.io>

Authentication

The screenshot shows a web browser's developer tools interface with the 'Application' tab selected. The left pane displays a login form with the following elements:

- Username* (text input)
- Password* (password input)
- Login (button)
- [Signup](#) (link)

The right pane shows a table of cookies. The table has the following columns: Name, Value, Domain, Path, Expires / Max-Age, Size, HttpOnly, Secure, SameSite, and Priority. One cookie is listed:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Priority
csrftoken	E9WpWRY...	127.0...	/	2021-07-16T11:...	73			Lax	Medium

The left sidebar of the developer tools shows the following categories and items:

- Application
 - Manifest
 - Service Workers
 - Clear storage
- Storage
 - Local Storage
 - Session Storage
 - IndexedDB
 - Web SQL
 - Cookies
 - http://127.0.0.1:8000
- Cache
 - Cache Storage
 - Application Cache
- Background Services
 - Background Fetch
 - Background Sync
 - Notifications
 - Payment Handler
 - Periodic Background Sync
 - Push Messaging

Authentication

The screenshot shows a web browser window with a login page. The page has a light orange background and contains the text "Hello johndoe123". Below this text, there are three bullet points with links: "Did you enjoy class today?", "What is your favourite course?", and "Who is your favourite teacher?". At the bottom left of the page, there is a "Logout" link.

The Chrome DevTools Application tab is open, showing the following sections:

- Application**
 - Manifest
 - Service Workers
 - Clear storage
- Storage**
 - Local Storage
 - Session Storage
 - IndexedDB
 - Web SQL
 - Cookies
 - http://127.0.0.1:8000
- Cache**
 - Cache Storage
 - Application Cache
- Background Services**
 - Background Fetch
 - Background Sync
 - Notifications
 - Payment Handler
 - Periodic Background Sync
 - Push Messaging

The **Application** tab also displays a table of cookies:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpO...	Secure	SameSite	Priority
sessionid	6euynixft2...	127.0...	/	2020-07-31T11:...	41	✓		Lax	Medium
csrftoken	ZalwArm4...	127.0...	/	2021-07-16T11:...	73			Lax	Medium