

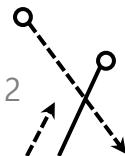
로그인 페이지 구현하기 1

DB&AI Lab.

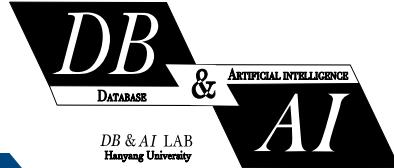
한양대학교 인공지능학과

목차

1. 프로젝트 폴더 생성
2. DB 구축
3. Backend API 구현
4. Next.js Frontend 설정
5. Next.js Frontend 구현



프로젝트 폴더 생성



- 이전과 동일한 가상환경 사용
- 프로젝트 루트 디렉터리 생성하기
 - 원하는 경로에 폴더 생성 후 폴더로 이동
 - ▶ 자료는 Desktop 아래에 login_project 이름으로 루트 폴더 생성
 - 방금 생성한 폴더 내에서 장고 프로젝트 생성하기

```
[oracle@localhost login_project]$ django-admin startproject config .
```

- Settings.py의 DATABASES 설정
 - 이전에 사용했던 django_project의 config/settings.py에서 DATABASES 내용 복사

- ▶ VirtualBox 예시:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.oracle',
        'NAME': 'localhost:1521/FREEPDB1',
        'USER': 'SYSTEM',
        'PASSWORD': 'oracle',
    }
}
```

프로젝트 폴더 생성(cont'd)

- config/settings.py 설정

```
INSTALLED_APPS = [
    'rest_framework',
    'corsheaders',
    'login.apps.LoginConfig',
```

- INSTALLED_APPS에 3개 추가

```
CORS_ALLOWED_ORIGINS= [
    "http://localhost:3000",
    "http://127.0.0.1:3000",
]
```

- CORS_ALLOWED_ORIGINS 생성 후 3000 포트 추가

```
MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
```

- 미들웨어에 cors 미들웨어 추가
- django-admin startapp login 명령어로 login앱 생성

DB 구축

- login/models.py 설정
 - user_id를 기본키로 하는 학생 테이블 생성

```
from django.db import models

class Student(models.Model):
    user_id = models.CharField(max_length=50, primary_key=True)
    password = models.CharField(max_length=50)
    student_no = models.CharField(max_length=20)

    class Meta:
        db_table = 'student'
```

- 아래 명령어를 통해 마이그레이션
 - python manage.py makemigrations
 - python manage.py migrate

Backend API 구현

- login/serializers.py 생성 및 작성

```
from rest_framework import serializers
from .models import Student

class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = ['user_id', 'password', 'student_no']
```

- 학생 테이블을 관리할 serializer 생성

Backend API 구현(cont'd)

- login/views.py 작성
 - 회원가입 API

```

from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import Student
from .serializers import StudentSerializer

class SignupAPI(APIView):
    def post(self, request):
        serializer = StudentSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
        return Response({"message": "회원가입 성공!", "data": serializer.data}, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
  
```

- post를 처리하는 회원가입 API 생성
- 생성 조건을 만족하면 save()를 통해 DB에 저장

Backend API 구현(cont'd)

- login/views.py 작성
 - 로그인 API

```
class LoginAPI(APIView):
    def post(self, request):
        user_id = request.data.get('user_id')
        password = request.data.get('password')

        try:
            student = Student.objects.get(user_id=user_id, password=password)
            serializer = StudentSerializer(student)
            return Response({
                "message": "로그인 성공",
                "user_id": student.user_id,
                "student_no": student.student_no
            }, status=status.HTTP_200_OK)
        except Student.DoesNotExist:
            return Response({"error": "아이디 또는 비밀번호가 틀렸습니다."}, status=status.HTTP_400_BAD_REQUEST)
```

- post를 처리하는 로그인API 생성
- post 정보를 이용해 DB 확인
 - 사용자 정보가 있으면 아이디와 학번을 반환
 - 없으면 error 반환

Backend API 구현(cont'd)

- URL 연결
 - login/urls.py 생성

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

urlpatterns = [
    path('signup/', views.SignupAPI.as_view(), name='signup'),
    path('', views.LoginAPI.as_view(), name='login'),
]
```

- signup과 login으로 분할

Backend API 구현(cont'd)

- URL 연결
 - config/urls.py

```
from django.contrib import admin
from django.urls import path, include

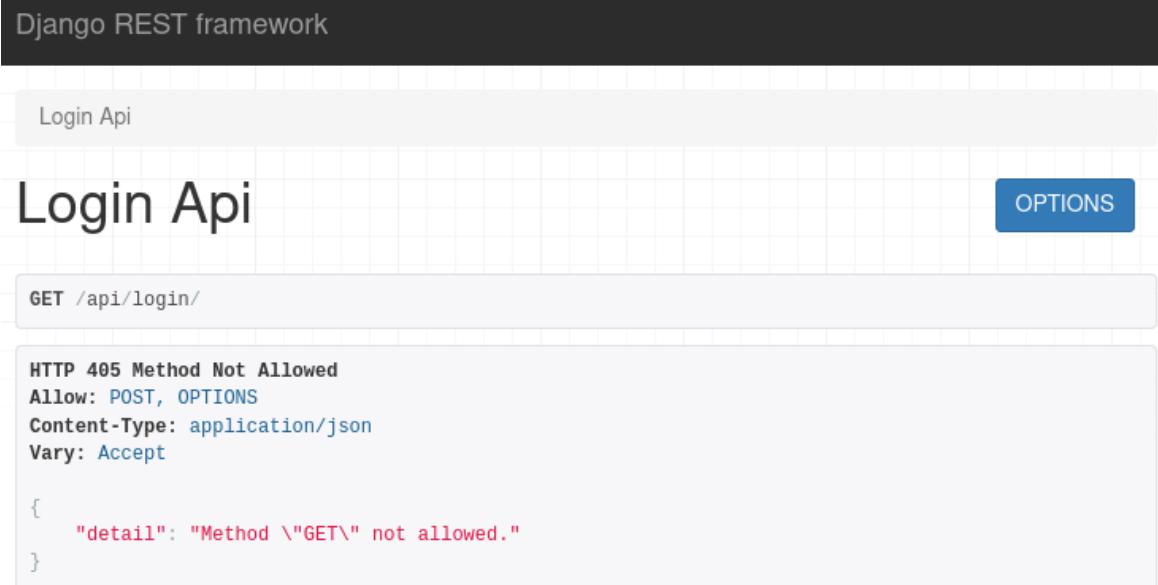
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/login/', include('login.urls')),
]
```

- login 로그인은 login/urls.py에서 처리되도록 경로 연결

Backend API 구현(cont'd)

- 중간 점검

- login_project 폴더에서 python manage.py runserver을 통해 백엔드 실행
- <http://127.0.0.1:8000/api/login/> 접속



- 링크를 통한 접속은 GET Method
 - ▶ GET 요청은 허용하지 않았으므로 에러

Next.js Frontend 설정

- Next.js 프로젝트 생성

- 프로젝트 루트 폴더(config 폴더가 보이는 곳)로 이동
- npx create-next-app@latest frontend로 Next.js 생성

```
? Would you like to use the recommended Next.js defaults? » - Use arrow-keys. Return to submit.
> Yes, use recommended defaults - TypeScript, ESLint, Tailwind CSS, App Router
  No, reuse previous settings
  No, customize settings
```

- ▶ Yes, use recommended defaults 선택
- ▶ 설치가 완료되면 frontend 폴더로 이동
 - cd frontend
- ▶ frontend 폴더 안에서 axios 설치
 - npm install axios
 - 설치 확인
 - frontend/package.json의 dependencies에 axios 있는지 확인

```
"dependencies": {
  "axios": "^1.13.2",
```

Next.js Frontend 설정(cont'd)

- Next.js 프로젝트 생성 확인
 - frontend 폴더에서 npm run dev으로 프론트엔드 서버 실행



To get started, edit the
page.tsx file.

Looking for a starting point or more instructions?

Head over to [Templates](#) or the [Learning](#) center.

- localhost:3000에서 Next.js 기본 화면 보이는지 확인

Next.js Frontend 구현

- frontend/app에 page.js 생성

```

"use client";
import { useState } from 'react';
import axios from 'axios';
import Link from 'next/link';

export default function LoginPage() {
  const [formData, setFormData] = useState({ user_id: '', password: '' });
  const [userInfo, setUserInfo] = useState(null); // 로그인 성공 시 정보 저장

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post('http://127.0.0.1:8000/api/login/', formData);
      // 로그인 성공 시 받아온 데이터를 상태에 저장
      setUserInfo(response.data);
    } catch (error){
      alert('로그인 실패 : 아이디와 비밀번호를 확인하세요 .');
    }
  };
}

```

Next.js Frontend 구현

- frontend/app에 page.js 생성

```
"use client";
import { useState } from 'react';
import axios from 'axios';
import Link from 'next/link';

export default function LoginPage() {
  const [formData, setFormData] = useState({ user_id: "", password: "" });
  const [userInfo, setUserInfo] = useState(null); // 로그인 성공 시 정보 저장

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post('http://127.0.0.1:8000/api/login/', formData);
      // 로그인 성공 시 받아온 데이터를 상태에 저장
      setUserInfo(response.data);
    } catch (error){
      alert('로그인 실패: 아이디와 비밀번호를 확인하세요.');
    }
  };
}
```

Next.js Frontend 구현

- frontend/app/page.js에 이어서 작성

```
// 로그인이 성공하면 보여줄 화면
if (userInfo) {
  return (
    <div style={{ padding: '20px', border: '2px solid blue' }}>
      <h1>로그인 성공! (실습 완료)</h1>
      <h2>환영합니다, {userInfo.user_id}님</h2>
      <h3>당신의 학번은 : {userInfo.student_no}입니다.</h3>
      <button onClick={() => window.location.reload()}>로그아웃 (새로고침)</button>
    </div>
  );
}

// 로그인 전 화면
return (
  <div style={{ padding: '20px' }}>
    <h1>로그인 페이지</h1>
    <form onSubmit={handleLogin}>
      <p><input name="user_id" placeholder="아이디" onChange={handleChange} required /></p>
      <p><input name="password" type="password" placeholder="비밀번호" onChange={handleChange} required /></p>
      <button type="submit">로그인</button>
    </form>
    <br />
    <Link href="/signup">
      <button>회원가입 하러 가기</button>
    </Link>
  </div>
);
}
```

Next.js Frontend 구현

- frontend/app/page.js에 이어서 작성

```
if (userInfo) {
  return (
    <div style={{ padding: '20px', border: '2px solid blue' }}>
      <h1>로그인 성공! (실습 완료)</h1>
      <h2>환영합니다, {userInfo.user_id}님</h2>
      <h3>당신의 학번은: {userInfo.student_no} 입니다.</h3>
      <button onClick={() => window.location.reload()}>로그아웃 (새로고침)</button>
    </div>
  );
}

return (
  <div style={{ padding: '20px' }}>
    <h1>로그인 페이지</h1>
    <form onSubmit={handleLogin}>
      <p><input name="user_id" placeholder="아이디" onChange={handleChange} required /></p>
      <p><input name="password" type="password" placeholder="비밀번호" onChange={handleChange} required /></p>
      <button type="submit">로그인</button>
    </form>
    <br />
    <Link href="/signup">
      <button>회원가입 하러가기</button>
    </Link>
  </div>
);
```

Next.js Frontend 구현

- frontend/app/에 signup 폴더 생성
- 폴더 생성 후 signup 폴더로 이동해서 page.js 생성

```
"use client";
import { useState } from 'react';
import axios from 'axios';
import { useRouter } from 'next/navigation';

export default function SignupPage() {
  const [formData, setFormData] = useState({
    user_id: '',
    password: '',
    student_no: ''
  });
  const router = useRouter();

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };
}
```

Next.js Frontend 구현

- frontend/app/signup에 page.js 생성

```
"use client";
import { useState } from 'react';
import axios from 'axios';
import { useRouter } from 'next/navigation';

export default function SignupPage() {
  const [formData, setFormData] = useState({
    user_id: '',
    password: '',
    student_no: ''
  });
  const router = useRouter();

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };
}
```

Next.js Frontend 구현

- frontend/app/signup/page.js에 이어서 작성

```

const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    // Django API로 데이터 전송
    await axios.post('http://127.0.0.1:8000/api/login/signup/', formData);
    alert('회원가입 완료! 로그인 페이지로 이동합니다.');
    router.push('/'); // 메인(로그인) 페이지로 이동
  } catch (error) {
    alert('회원가입 실패: 아이디가 중복되었거나 오류가 발생했습니다.');
  }
}
return (
  <div style={{ padding: '20px' }}>
    <h1>회원가입 페이지 (실습 확인용)</h1>
    <form onSubmit={handleSubmit}>
      <p><input name="user_id" placeholder="아이디" onChange={handleChange} required /></p>
      <p><input name="password" type="password" placeholder="비밀번호" onChange={handleChange} required /></p>
      <p><input name="student_no" placeholder="학번 (예: 20240001)" onChange={handleChange} required /></p>
      <button type="submit">가입하기</button>
    </form>
  </div>
);
}

```

Next.js Frontend 구현

- frontend/app/signup/page.js에 이어서 작성

```

const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    // Django API로 데이터 전송
    await axios.post('http://127.0.0.1:8000/api/login/signup/', formData);
    alert('회원가입 완료! 로그인 페이지로 이동합니다.');
    router.push('/'); // 메인(로그인) 페이지로 이동
  } catch (error) {
    alert('회원가입 실패: 아이디가 중복되었거나 오류가 발생했습니다.');
  }
};

return (
  <div style={{ padding: '20px' }}>
    <h1>회원가입 페이지 (실습확인용)</h1>
    <form onSubmit={handleSubmit}>
      <p><input name="user_id" placeholder="아이디" onChange={handleChange} required /></p>
      <p><input name="password" type="password" placeholder="비밀번호" onChange={handleChange} required /></p>
      <p><input name="student_no" placeholder="학번 (예: 20240001)" onChange={handleChange} required /></p>
      <button type="submit">가입하기</button>
    </form>
  </div>
);

```

Q&A



THANK YOU