

Facade

دیزاین پترن هادر React

Structural Patterns

به زبان ساده همراه با مثال های عملی
کاربرد دیزاین پترن ها برای برنامه نویسان React



Structural Patterns

Facade

الگوی Facade می‌تواند یک رابط ساده برای کار با سیستم‌های پیچیده ارائه دهد. از این الگو می‌توان در جاهایی مانند ساده سازی تعاملات با سرویس‌های یا منطق‌های پیچیده استفاده کرد.

فرض کنید می‌خواهیم یک رابط برای مدیریت ورود، خروج و بررسی وضعیت احراز هویت کاربر ایجاد کنیم که پیچیدگی‌های داخلی آن پنهان باشد.

بخش IAuthResponse

ابتدا یک Interface به شکل زیر برای Response ایجاد می کنیم.



IAuthResponse



```
export interface IAuthResponse {  
    success: boolean;  
    token?: string;  
    message?: string;  
}
```

بخش AuthService

تمامی های لازم همراه با پیچیدگی ها و منطق های مورد نیاز برای کار با API بخش را در این سرویس من نویسیم.



```
import { IAuthResponse } from './auth-response.interface';

export class AuthService {
  async login(username: string, password: string): Promise<IAuthResponse> {
    try {
      const response = await fetch('/api/login', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ username, password }),
      });

      if (!response.ok) {
        throw new Error('Login failed');
      }

      const data = await response.json();
      return { success: true, token: data.token };
    } catch (error) {
      let message = 'An unknown error occurred';
      if (error instanceof Error) {
        message = error.message;
      }
      return { success: false, message: message };
    }
  }

  async logout(): Promise<void> {
    // Clear session or make an API call
    console.log('User logged out');
  }

  async isAuthenticated(): Promise<boolean> {
    // Check local storage or make an API call
    const token = localStorage.getItem('authToken');
    return !!token;
  }
}
```

login()

logout()

isAuthenticated()

AuthService

بخش useAuth

در این بخش یک Hook را به شکل زیر می سازیم تا پیچیدگی ها را در Component به حداقل برسانیم.



useAuth

```
import { useState } from 'react';
import { AuthService } from './auth.service';
import { IAuthResponse } from './auth-response.interface';

const authFacade = new AuthService();

export function useAuth() {
  const [isAuthenticated, setIsAuthenticated] = useState(false);

  const login = async (username: string, password: string): Promise<IAuthResponse> => {
    const response = await authFacade.login(username, password);
    if (response.success && response.token) {
      localStorage.setItem('authToken', response.token);
      setIsAuthenticated(true);
    }
    return response;
  };

  const logout = async (): Promise<void> => {
    await authFacade.logout();
    localStorage.removeItem('authToken');
    setIsAuthenticated(false);
  };

  const checkAuth = async (): Promise<void> => {
    const authenticated = await authFacade.isAuthenticated();
    setIsAuthenticated(authenticated);
  };

  return { isAuthenticated, login, logout, checkAuth };
}
```

بخش LoginForm

در این بخش مد نظر خود را به شکل زیر می سازیم.



```
●●●

import { useState } from 'react';
import { useAuth } from './pattern/useAuth.hook';

const LoginForm: React.FC = () => {
  const { login, isAuthenticated } = useAuth();
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState<string | null>(null);

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setError(null);

    const response = await login(username, password);
    if (!response.success) {
      setError(response.message || 'An error occurred');
    }
  };

  return (
    <div>
      <form onSubmit={handleSubmit}>
        <div>
          <label> Username: </label>
          <input type='text' value={username} onChange={(e) => setUsername(e.target.value)} />
        </div>
        <div>
          <label> Password: </label>
          <input type='password' value={password} onChange={(e) => setPassword(e.target.value)} />
        </div>
        <button type='submit'>Login</button>
        {error && <p style={{ color: 'red' }}>{error}</p>}
      </form>
    </div>
  );
}

export default LoginForm;
```

پیچیدگی و منطق را در
به حداقل رسانده ایم.

```
●●●

<div>
  {isAuthenticated ? (
    <p>You are logged in!</p>
  ) : (
    <form onSubmit={handleSubmit}>
      <div>
        <label> Username: </label>
        <input type='text' value={username} onChange={(e) => setUsername(e.target.value)} />
      </div>
      <div>
        <label> Password: </label>
        <input type='password' value={password} onChange={(e) => setPassword(e.target.value)} />
      </div>
      <button type='submit'>Login</button>
      {error && <p style={{ color: 'red' }}>{error}</p>}
    </form>
  )}
</div>
```

طرز استفاده از

در پایان به شکل زیر از LoginForm که ساختیم، استفاده می کنیم.



FacadeDesignPattern

```
import LoginForm from './LoginForm';

const FacadeDesignPattern = () => {
  return (
    <div>
      <LoginForm />
    </div>
  );
};

export default FacadeDesignPattern;
```