Next-auth

Permet de gérer l'authentification sur une application nextJS en utilisant différent Provider. A l'avantage d'utiliser la session nextJS.

Installation

```
npm install next-auth@beta @types/next-auth
```

Configuration

```
import NextAuth from "next-auth";
import CredentialsProvider from "next-auth/providers/credentials";
import HttpService from "@/services/HttpService";
import API_URLS from "@/constants/ApiUrls";
import { decodeJwt } from "jose";

interface AuthToken {
    accessToken: string;
    refreshToken: string;
    user: any;
}

async function refreshAccessToken(token: AuthToken) {
    try {
        const refreshedToken = await HttpService.post(API_URLS.refreshToken, {
            refreshToken: token.refreshToken,
        });

        if (!refreshedToken || !refreshedToken.accessToken) {
            throw new Error("Refresh token failed");
        }

        return {
            ...token,
            accessToken: refreshedToken.accessToken,
            accessTokenExpires: Date.now() + refreshedToken.expiresIn * 1000,
            refreshToken: refreshedToken.refreshToken ?? token.refreshToken,
        };
    } catch (error) {
        console.error("Erreur lors du rafraîchissement du token", error);
        return { ...token, error: "RefreshAccessTokenError" };
    }
}

// @ts-ignore
export const authConfig = {
```

```
    pages: {
        signIn: '/login',
        signOut: "/logout",
    },
    providers: [
        CredentialsProvider({
            id: "credentials",
            name: "Credentials",
            credentials: {
                username: { label: "Login", type: "text" },
                password: { label: "Password", type: "password" },
            },
            async authorize(credentials) {
                const resp = await HttpService.post(API_URLS.authLogin, {
                    username: credentials.username,
                    password: credentials.password,
                });

                if (resp.ok) {
                    throw new Error("Invalid credentials");
                }
                const token=resp.data;
                const user = decodeJwt(token.accessToken);

                return {
                    id: user.sub,
                    name: user.name,
                    email: user.email,
                    role: user.role,
                    accessToken: token.accessToken,
                    //refreshToken: token.refreshToken,
                    accessTokenExpires: Date.now() + token.expiresIn * 1000,
                };
            },
        }),
    ],
    callbacks: {
        async jwt({ token, user }) {
            if (user) {
                return {
                    accessToken: user.accessToken,
                    //refreshToken: user.refreshToken,
                    accessTokenExpires: Date.now() + 1000 * 60 * 60, // 1h
                    user: {
                        id: user.id,
                        name: user.name,
                        email: user.email,
                        role: user.role
                    },
                };
            }

            if (Date.now() < token.accessTokenExpires) {
                return token;
            }
```

```
            return refreshAccessToken(token);
        },
        async session({ session, token }) {
            console.log("session in session method", session);
            console.log("token in session method", token);
            return {
                ...session,
                user: {
                    ...session.user,
                    id: token.user.id,
                    name: token.user.name,
                    email: token.user.email,
                    accessToken: token.accessToken,
                    //refreshToken: token.refreshToken,
                },
            };
        },
    },
    session: {
        strategy: 'jwt',
    },
    secret: process.env.NEXTAUTH_SECRET as string,
} satisfies NextAuthConfig;

//@ts-ignore
export const {handlers, auth, signIn, signOut} = NextAuth(authConfig);
```

Route api

Toutes les requêtes NextAuth passeront par cette route : `src/app/api/auth/[…nextauth]/route.tsx`

```
import {handlers} from "@/auth";
export const {GET, POST} = handlers;
```

Lib

Fichier utilitaire pour simplifier connexion et déconnexion dans `src/lib/actions.ts`:

```
'use server';
import {signIn, signOut} from "@/auth";


interface LoginFormValues {
    username: string;
    password: string;
    rememberMe?: boolean;
};
export const submitLogin = async (formData: LoginFormValues): Promise<any> => {
```

```
    await signIn('credentials', {...formData, redirectTo: '/', redirect: true});
};

export const submitLogout = async () => {
    await signOut({redirect: true, redirectTo: '/logout'});
};
```

LoginForm

Exemple de composant pour le login :

```
'use client';
import {Button, Checkbox, Form, Input} from "antd";
import {submitLogin} from "@/lib/actions";


export default function SignInComponent(){
    const onFinish = async (values: any) => {
        await submitLogin(values);
    };

    const onFinishFailed = (errorInfo: any) => {
        console.log('Failed:', errorInfo);
    };
    return
     <div>
         <Form
             name="basic"
             labelCol={{ span: 8 }}
             wrapperCol={{ span: 16 }}
             style={{ maxWidth: 600 }}
             initialValues={{ remember: true }}
             onFinish={onFinish}
             onFinishFailed={onFinishFailed}
             autoComplete="off"
         >
             <Form.Item
                 label="Username"
                 name="username"
                 rules={[{ required: true, message: 'Please input your username!'
}]}
             >
                 <Input />
             </Form.Item>

             <Form.Item
                 label="Password"
                 name="password"
                 rules={[{ required: true, message: 'Please input your password!'
}]}
             >
                 <Input.Password />
             </Form.Item>
```

```
            <Form.Item name="remember" valuePropName="checked" label={null}>
                <Checkbox>Remember me</Checkbox>
            </Form.Item>

            <Form.Item label={null}>
                <Button type="primary" htmlType="submit">
                    Submit
                </Button>
            </Form.Item>
        </Form>
    </div>
}
```

Configuration

Dans le fichier `.env` ou `.env.local`, définir la variable NEXTAUTH_URL:

```
NEXTAUTH_URL=http://127.0.0.1:3000
```

Génération du secret pour encodage/décodage JWT :

```
npx auth secret
```

Le **secret** est généré et inséré dans le fichier `.env.local`.

Usage

Accès à la session nextAuth :

Côté serveur

```
import {auth} from "@/auth";

const session= await auth();
console.log(session.user); //user connecté
```

Côté client

```
'use client';
import {useSession} from "next-auth/react";
```

```
const userData=useSession();
console.log(userData.data?.user); //user connecté
```

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/web/framework/nextjs/nextauth

Last update: **2025/08/12 02:35**