

# 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 :

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

From:

<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:

<http://slamwiki2.kobject.net/web/framework/nextjs/nextauth?rev=1742371543>

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

