EUWA Wrapper Interface

EUWA Wrapper Interface (også kaldet *wrapper* nedenfor) giver et enkelt indgangs- og kommunikationspunkt til slutbrugerens webapplikationer (EUWA). Det giver udviklerne mulighed for at starte EUWA hurtigt og kommunikere med dem uden at administrere indlæsningen af applikationerne, f.eks. Hvis du vil starte chat med en brugerdefineret knap eller tilføje konfiguration / variabler (runtime) fra dit websted.

Pakken offentliggøres i vores offentlige NPM-register som @puzzel/euwa-wrapper

Interface

declare interface Config { customerKey: string, configId: string } declare interface Options { settings: ApplicationSettings, hooks: Hooks } declare interface ApplicationList { [app: string]: string } declare interface Hooks { [hook: string]: Function } declare interface ApplicationBridge { api: ApplicationAPI, publish: (event: string, ...data: any) => void, subscribe: (event: string, callback: Function) => void, } declare interface ApplicationSettings { [app: string]: object } declare interface ApplicationAPI { [method: string]: Function } declare class EUWA { static APPS: ApplicationList constructor({customerKey, configId}: Config, {settings, hooks}: Options); getApplication(id: string): Promise; getApplicationBeforeLoad(id: string): ApplicationBridge; }

Opretter forbindelse til NPM-registreringsdatabasen

En .npmrc fil skal oprettes enten i dit projekt eller på brugerniveau. Læs mere om .npmrc på https://docs.npmjs.com/cli/v6/configuring-npm/npmrc

Følgende linjer skal tilføjes:

@puzzel:registry=https://puzzel.pkgs.visualstudio.com/public/_packaging/main/npm/registry/ always-auth=true

Grundlæggende brug

Den grundlæggende brug indlæser EUWA med konfigurationssæt fra Puzzels Administration Portal.

import { EUWA } from '@puzzel/euwa-wrapper'; new EUWA({ configId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx', customerKey: 123456 });

API-brug

Instantiering af EUWA-klassen vil returnere en simpel API, der giver adgang til enhver slutbruger-webapplikationskontekst.

import {EUWA} from '@puzzel/euwa-wrapper'; const euwa = new EUWA({ configId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx', customerKey: 123456 }); // Subscribe to click event on your start button document.querySelector('#your-start-button').addEventListener('click', async () => { // Get the chat application context const chat = await euwa.getApplication(EUWA.APPS.CHAT); // Use the Chat's API to retrieve it's state const state = chat.api.getState(); // Start a chat, if the user is not already in session if (!state.isConnected) { chat.api.startChat(); } });

Brug af kroge

onBeforeLoad
Dette giver dig mulighed for at abonnere på begivenheder eller udføre andre handlinger, inden applikationerne indlæses. Da de forskellige applikations API'er er defineret af selve applikationen, vil de imidlertid ikke være tilgængelige - kun offentliggør / abonner-interface er tilgængelig.

> Note
>
> EUWA's getApplicationBeforeLoad-metode er specielt designet til at blive brugt specifikt med denne krog. Det venter ikke på applikationens belastning og returnerer det grundlæggende kommunikationsinterface - begivenheder.

import {EUWA} from '@puzzel/euwa-wrapper'; const euwa = new EUWA({ configId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx', customerKey: 123456 }, { hooks: { // All hooks accept functions onBeforeLoad: subscribeToChatInit } }); function subscribeToChatInit() { // Get the Chat's event interface const chat

= euwa.getApplicationBeforeLoad(EUWA.APPS.CHAT); // Subscribe to chatInit* event chat.subscribe('chatInit', data => { console.log('Chat Init Data:', data); }); }

** En komplet liste over begivenheder kan findes i vores Chat Front End API-artikel.*

Tilsidesættelse af indstillinger

import {EUWA} from '@puzzel/euwa-wrapper'; const euwa = new EUWA({ configId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx', customerKey: 123456 }, { settings: { // Application name as first-level property [EUWA.APPS.CHAT]: { // Properties names as listed in Chat Admin showForm: false } } });