EUWA Wrapper Interface

EUWA Wrapper Interface (även kallat *wrapper* nedan) ger en enda in- och utgångspunkt för slutanvändarwebapplikationer (EUWA). Det låter utvecklarna initiera EUWA snabbt och kommunicera med dem utan att hantera laddningen av applikationerna, t.ex. om du vill börja chatta med en anpassad knapp eller lägga till konfiguration / variabler (runtime) från din webbplats.

Paketet publiceras i vårt offentliga NPM-register som `@puzzel/euwa-wrapper`

Gränssnitt

```
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; }
```

Ansluter till NPM-registret

En `.npmrc` fil ska skapas antingen i ditt projekt eller på användarnivå. Läs mer om .npmrc på https://docs.npmjs.com/cli/v6/configuring-npm/npmrc

Följande rader bör läggas till:

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

Grundläggande användning

Den grundläggande användningen laddar `EUWA` , med konfigurationsuppsättning från Puzzels administrationsportal.

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

API-användning

Instabiliseringen av EUWA-klassen kommer att returnera ett enkelt API som gör det möjligt att få tillgång till alla slutanvändares webbapplikationskontext.

```
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(); } });
```

Använda krokar

onBeforeLoad
Detta gör att du kan prenumerera på händelser eller göra andra åtgärder innan applikationerna laddas. Eftersom de olika programmens API: er definieras av själva applikationen kommer de dock inte att finnas tillgängliga - endast gränssnittet publicera / prenumerera är tillgängligt.

> Note
>
> EUWA: s getApplicationBeforeLoad-metod är speciellt utformad för att användas med specifikt denna krok. Det väntar inte på applikationens laddning och returnerar det grundläggande kommunikationsgränssnittet - händelser.

```
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 fullständig lista över händelser finns i vår artikel om Chat Front End API.*

Åsidosättande inställningar

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 } } });