Oversikt over Widget API

Bakgrunn

Widget-API-et er utformet med disse målene for øyet: – for å skjule kompleksiteten til agentapplikasjonen for widgetene – for å presentere et lite og konsistent grensesnitt for widgetene – for å opprettholde et stabilt grensesnitt for widgetene

Oversikt

Widget-API-et er delt i to hoveddelsystemer: hendelsessystemet og grensesnittene. Hendelsessystemet brukes til kringkasting av ulike meldinger fra tjenestene til widgetene og til widget-til-widget-kommunikasjon. Grensesnittene er en fasade til tjenestene og kjernefunksjonaliteten.

Widgetmeldinger

Som tidligere nevnt ligger de eksterne widgetene vanligvis på et annet domene, og de har ikke direkte tilgang til widget-API-et. For å omgå domenebarrieren åpner agentapplikasjonens kjerne en meldingskanal med nettleserens meldings-API.

Koden nedenfor viser én måte å abonnere på og håndtere meldinger fra API-et på:

```
const origin = 'the origin of the agent application';
let port;
window.addEventListener('message', message => {
      // Make sure that the channel comes from the correct source:
      if (message.origin !== origin) return;

     // Setup the communication channel:
     if (!port) { port = message.ports[0]; port.onmessage = receiver; } });

function receiver(message) {
      //code here
}
```

Meldingenes nyttelast befinner seg imessage.data. Kjernen legger til en message.data.type-egenskap til alle meldinger også.

Forespørsler til widget-API-et kan bare sendes gjennom den angitte porten:

```
port.postMessage(message);
```

Grensesnitt

Hvis du vil hente en egenskap eller kalle en metode i widget-API-et, må du bruke meldingsformatet {call, args} , der call er banen til metoden (eller egenskapen) i API-et.

Hvis du skal kalle en metode, er args en matrise med alle nødvendige argumenter for metodekallet. Eksempel:

```
port.postMessage({
      call: 'tab.setTitle',
     args: ['new title']
});
```

Hvis metoden returnerer et resultat, sendes dette til den eksterne widgeten av port.onmessage-behandleren i formatet {name, value, type}, der name er navnet på den forespurte egenskapen eller metoden, value verdien for egenskapen eller resultatet av kallet, og type strengens «resultat».

Eksempel på svar på et getOption-kall:

```
{
        name: 'widget.getOption',
        value: 'https://demo.puzzel.com/dev/widgets/external/demo/',
```

```
        type: 'result'
}
```

Merk at på grunn av måten meldings-API-et fungerer på, befinner nyttelasten for meldingen seg i egenskapen message.data.

Hvis den kalte metoden ikke returnerer et resultat, sender ikke kjernen noen melding. Hvis den kalte metoden returnerer et løfte, sendes meldingen når løftet løses eller avvises. Hvis løftet løses, sender kjernen en standard resultatmelding, der value vil inneholde verdien av løftet. Hvis løftet avvises, blir det sendt en feilmelding:

```
{
     name: 'widget.setOption',
     value: 'Unexpected end of JSON input',
    type: 'error'
}
```

Hvis det kreves samsvar mellom et kall og et resultat, kan det valgfrie id legges til i forespørselen. Detgir følgende:

```
{
   call: 'tab.getOption',
  args: ['option name'],
   id: '0123456789'
}
```

Resultat:

```
{
   name: 'tab.getOption',
   value: 'option value',
   id: '0123456789'
}
```

Widgeten kan også observere en egenskap for endringer ved å sende en {watch}-melding. Feltet watch skal inneholde banen til egenskapen i widget-API-et.

Hvis verdien for denne egenskapen endres, sender kjernen en melding med formen {name, old, new, type}, der name er den samme egenskapsbanen, old verdien av egenskapen før endringen, new verdien etter endringen og type den «endrede» strengen.

Hendelser

De eksterne widgetene kan abonnere på hendelser ved å sende en melding med formen {subcribe, options: {once, address}} til kjernen. Feltet subscribe må inneholde hendelsens navn. Hele options-feltet er valgfritt, og det samme gjelder dets egenskaper: det boolske once og address-strengen. Address har samme betydning som i metodene til ExtendedEventAggregator. Det angitte once betyr at subscribeOnce-metoden vil bli brukt, dvs. at den eksterne widgeten bare vil motta én enkelt hendelse før abonnementet avslutter seg selv.

Hendelsene vil bli mottatt med en melding med formen {name, value, type}, der name er navnet på hendelsen, value nyttelasten og type «hendelsen».

```
{
    name: 'userStatusChanged',
    value: 'System',
    type: 'event'
}
```

Den fullstendige API-referansen vil bli gjort tilgjengelig her

