Oversigt over Widget API

Baggrund

Widget API er designet med følgende mål for øje: at skjule kompleksiteten af agentapplikationen for widgeterne, at vise widgeterne en lille og ensartet grænseflade og at opretholde en stabil grænseflade for widgeterne.

Oversigt

Widget API er opdelt i to overordnede undersystemer: hændelsessystemet og grænsefladerne. Hændelsessystemet bruges til at sende forskellige beskeder fra tjenesterne til widgeterne og til indbyrdes kommunikation mellem widgeterne. Grænsefladerne udgør facaden mod tjenesterne og kernefunktionaliteten.

Widgetbeskeder

Som tidligere nævnt hostes de eksterne widgets generelt på et andet domæne og har ikke direkte adgang til Widget API. For at omgå domænebarrieren åbner agentapplikationens kerne en beskedkanal via browserens Messaging API.

Nedenstående kode viser én måde at abonnere på og håndtere beskeder fra API'et:

```
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 }
```

Beskedernes indhold ligger i message.data. Kernen føjer desuden egenskaben message.data.type til alle beskeder.

Anmodninger til Widget API kan kun sendes via den angivne port:

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

Grænseflader

Hvis du vil hente en egenskab eller kalde en metode i Widget API, skal du bruge beskedformatet {call, args}, hvor call er stien til metoden (eller egenskaben) i API'et.
Hvis du bruger et metodekald, er args et array, som indeholder alle påkrævede argumenter til metodekaldet. Eksempel:

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

Hvis metoden returnerer et resultat, sendes det til den eksterne widget af handleren port.onmessage i formatet {name, value, type}, hvor name er navnet på den anmodede egenskab eller metode,value er egenskabens værdi eller resultatet af kaldet, og type er streng-'resultatet'.
Eksempel på et svar på et getOption-kald:

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

Bemærk, at på grund af den måde, Messaging API fungerer på, ligger beskedens indhold i egenskaben message.data.

Hvis den kaldte metode ikke returnerer et resultat, sender kernen ikke en besked. Hvis den kaldte metode returnerer et tilsagn, sendes beskeden, når tilsagnet er løst eller afvist. Hvis tilsagnet er løst, sender kernen en standardresultatbesked, hvor value indeholder tilsagnets værdi. Hvis tilsagnet er afvist, vises en 'fejl'-besked:

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

Hvis det er påkrævet at matche et kald til et resultat, kan det valgfri id føjes til anmodningen. Det returneres igen:

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

Resultat:

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

Widgeten kan også undersøge en egenskab for ændringer ved at sende en {watch}-besked. Feltet watch skal indeholde stien til egenskaben i Widget API.

Hvis egenskabens værdi ændrer sig, sender kernen en {name, old, new, type}-besked, hvor name er den samme egenskabssti, old er egenskabens værdi før ændringen, new er værdien efter ændringen, og typethat er den streng, der er 'ændret'.

Hændelser

De eksterne widgets kan abonnere på hændelser ved at sende en {subscribe, options: {once, address}}-besked til kernen. Feltet subscribe skal indeholde hændelsens navn. Hele feltet options er valgfrit, og det er dets egenskaber også: det booleske once og address-strengen. Adressen har samme betydning som i ExtendedEventAggregators metoder. Sættet once betyder, at metoden subscribeOnce anvendes, dvs. den eksterne widget modtager kun en enkelt hændelse, før abonnementet automatisk opsiges.
Hændelserne modtages ved hjælp af en {name, value, type}-besked, hvor name er hændelsens navn, value er indholdet og type er 'hændelse'.

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

Den fuldstændige dokumentation til API'et er tilgængelig her

