Documentazione di Node-Red

Sviluppo di Flussi:

Best practice per la creazione di flussi chiari e riutilizzabili
Sviluppo di Flussi:

In qualsiasi linguaggio di programmazione, un aspetto fondamentale per creare codice di facile manutenzione è assicurarsi che sia anche ben documentato.
Una buona documentazione ha diversi scopi:

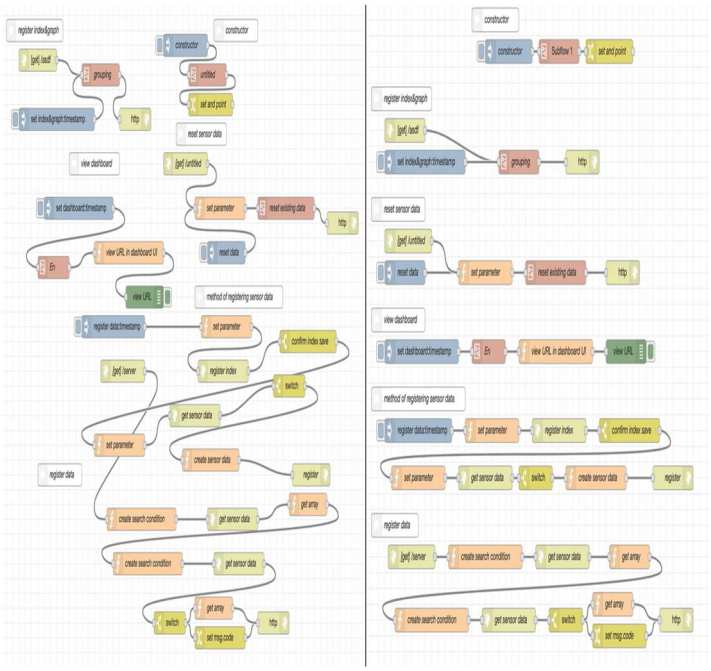
1. Sebbene tutto possa sembrare ovvio durante la creazione di un flusso, in futuro sarete grati a voi stessi per aver fornito una descrizione dettagliata quando ci tornerete in seguito.
2. Se si condivide un flusso con altri, questo li aiuterà a capire cosa fa e come funziona.
3. Se un flusso fornisce un'API esterna, si dovrà documentare come tale API dovrebbe essere utilizzata, ovvero quali proprietà o parametri sono previsti.
4. Quando si scrive la documentazione, l'atto di descriverne il comportamento potrebbe aiutare a identificare le parti che potrebbero essere migliorate.

In un ambiente di programmazione visuale come Node-RED, la documentazione può assumere diverse forme.

- I flussi possono essere letti nell'area di lavoro per visualizzare il flusso logico degli eventi. Si dovrebbe far sì che lo scopo di ciascun nodo sia facilmente identificabile e che siano ben disposti per ridurre al minimo il numero di collegamenti che si incrociano.
- I gruppi possono essere utilizzati per identificare sezioni distinte dei flussi.
- Spostare le parti di uso comune in sottoflussi può aiutare a ridurre la complessità visiva del flusso.
- È possibile aggiungere una documentazione più completa a livello di nodo, gruppo o scheda.

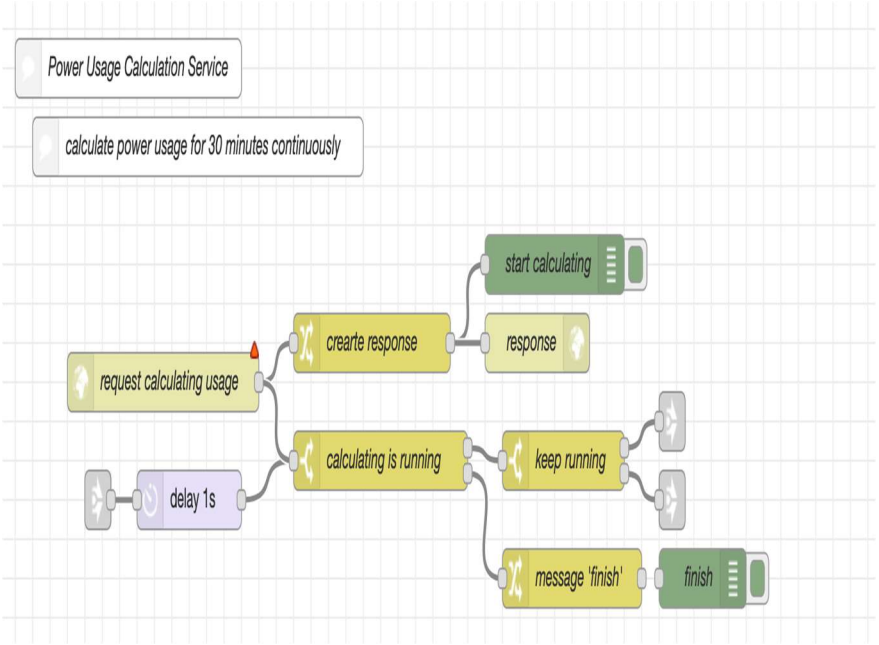
Disposizione dei Flussi

La sezione struttura del flusso di questa guida ha esaminato come disporre i componenti logici dei flussi. Questa sezione prende in considerazione l'aspetto visivo del layout del flusso.
L'obiettivo è semplificare la consultazione del flusso senza dover saltare da un'area di lavoro all'altra o seguire più fili che si incrociano e appaiono aggrovigliati.
L'approccio che garantisce la massima leggibilità è quello di mantenere ogni unità di elaborazione su un'unica linea orizzontale, ove possibile. Il comportamento predefinito dell'editor, che prevede l'aggancio dei nodi a una griglia sulla scheda, aiuta a mantenerli allineati.



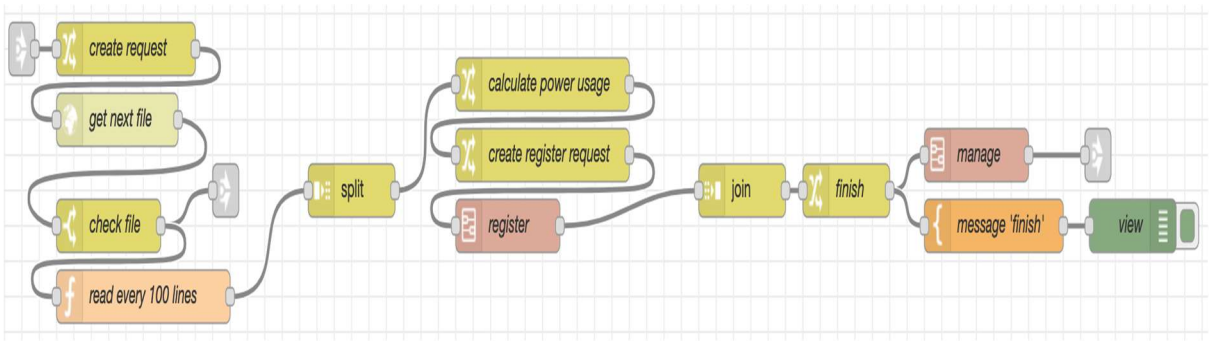
Aligning flows in horizontal rows

Se un nodo ha più di una porta di uscita, l'allineamento verticale del flusso ramificato semplifica il confronto e il contrasto dei flussi.



Aligning branched flows

Quando un flusso diventa troppo lungo, è possibile utilizzare con successo la disposizione verticale di alcuni nodi. Nella figura seguente, alcuni nodi sono disposti verticalmente per suggerire una relazione tra loro. È più facile comprendere la natura del flusso complessivo se è visivamente evidente da quali sezioni più piccole è composto e come queste si relazionano tra loro.



Vertically aligning logical segments of a long flow

In alcuni casi, queste sezioni più piccole potrebbero essere candidate per essere spostate in Subflow che ne ridurranno la complessità visiva. Ciò è particolarmente vero se quella sezione più piccola potesse essere riutilizzata altrove nei flussi.

Nomi dei Nodi

La maggior parte dei nodi ha una proprietà `name` utilizzabile per personalizzare l'etichetta visualizzata nell'area di lavoro. Questa proprietà dovrebbe essere utilizzata per etichettare correttamente i punti chiave di un flusso.

Ad esempio, se un nodo «Change» ha una singola regola che imposta `msg.payload` sull'ora corrente, la sua etichetta predefinita sarà `set msg.payload`. Questo aiuta in qualche modo, ma non rivela lo scopo completo del nodo. Un nome come `Get current time` sarebbe molto più chiaro.

C'è un compromesso da considerare. Più lunga è l'etichetta, maggiore è lo spazio di cui ha bisogno nel flusso. Più corta è l'etichetta, minori sono le informazioni che può condividere.

Per alcuni nodi, potrebbe essere opportuno nascondere completamente l'etichetta per ridurre al minimo lo spazio orizzontale che occupa nel flusso, lasciando più spazio agli altri nodi.

Oltre all'etichetta, i nodi possono anche avere un'icona personalizzata. Ad esempio, se si dispone di diversi nodi «MQTT In» per diversi tipi di dispositivi, potrebbe essere utile personalizzare l'icona in base al tipo di dispositivo. Questa operazione deve essere eseguita con attenzione, poiché l'icona è uno dei principali metodi per identificare il tipo di un particolare nodo.

La scelta di nomi appropriati per gli elementi si applica anche alle schede e ai Subflow utilizzati.

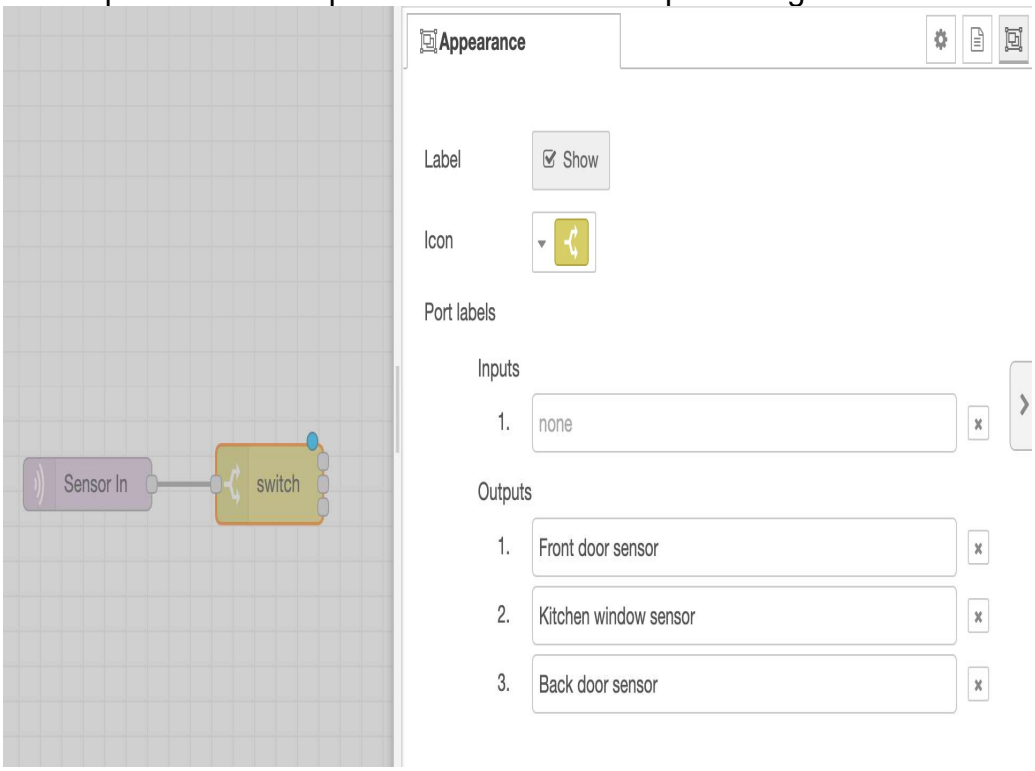
È molto importante anche per i nodi Link. Senza un nome impostato, è necessario utilizzare l'ID interno del nodo Link quando si creano collegamenti tra diverse schede. Ciò rende difficile identificare il nodo di destinazione corretto e possono verificarsi errori. Se si considera che i nodi Link forniscono API tra le diverse schede, sarà necessario scegliere uno schema di denominazione appropriato. I nomi dovrebbero identificare chiaramente il punto di inizio e di fine di ciascun flusso.

Aggiunta di Label alle Porte

Se un nodo ha più output, può essere difficile seguirne la logica se non è chiaro in base a quali condizioni un messaggio può essere inviato da un particolare output.

È qui che l'aggiunta di etichette alle porte può aiutare a documentare la logica prevista.

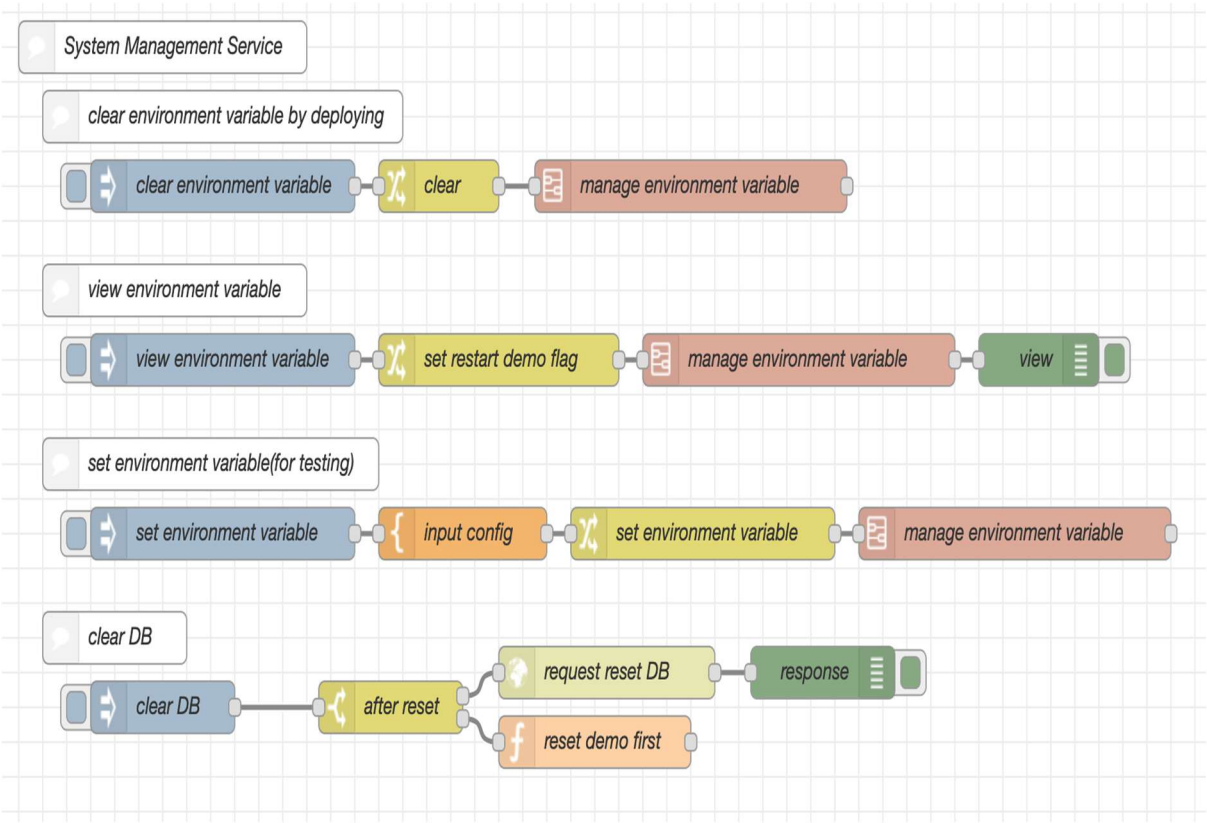
Ad esempio, il nodo Switch fornisce etichette di default per i suoi output, visualizzate quando il mouse vi passa sopra. Possono aiutare a identificare rapidamente lo scopo di ogni ramo del flusso.
Sebbene le etichette di default possano essere sufficienti nel contesto del flusso stesso, è anche possibile personalizzarle per fornire informazioni più dettagliate.



Custom output labels on the Switch node's Appearance tab

Commenti in Linea

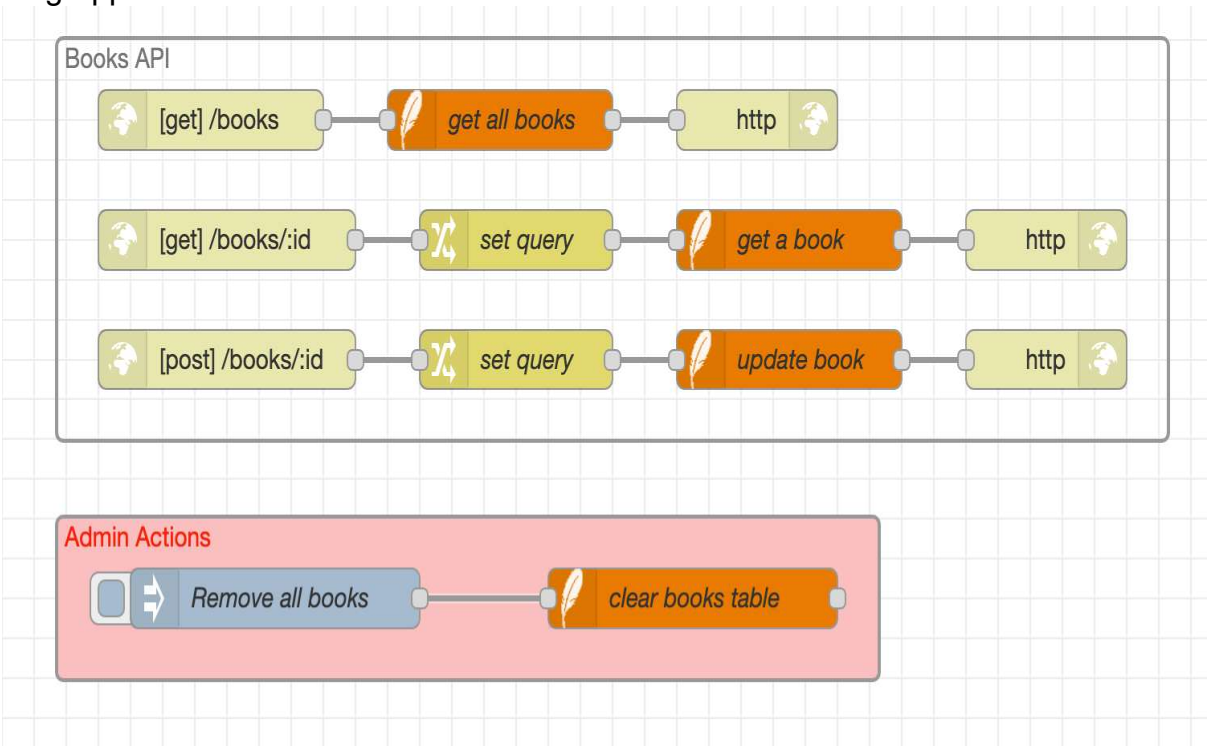
Il nodo «Comment» può essere utilizzato per aggiungere commenti in linea al flusso: sia l'etichetta del nodo, sia la sua descrizione, che verrà visualizzata nella barra laterale «Information» quando selezionata.
Indentando i flussi sulla pagina, è possibile indicare un raggruppamento implicito dei diversi componenti.



Documenting flows with the Comment node

Raggruppamento dei Nodi

Una disposizione più esplicita dei flussi può essere ottenuta raggruppando i nodi correlati. Il colore di sfondo di ciascun gruppo può essere utilizzato anche per evidenziare diversi tipi di gruppo.



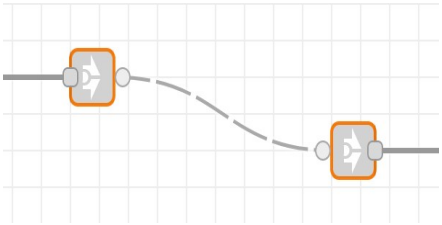
Grouping nodes

Aggiunta di Documentazione più Estesa

Tutte le tecniche discusse finora riguardano l'aspetto visivo dei flussi. Per aggiungere una documentazione più approfondita, è necessario qualcosa in più.
Per ogni nodo, gruppo e «Tab» [scheda] è possibile aggiungere una documentazione più estesa nel Tab Description nella finestra di dialogo di edit. Questa guida può essere formattata utilizzando il Markdown e includere elenchi, tabelle e link. Questa documentazione viene poi visualizzata nella Barra laterale Information quando l'elemento viene selezionato.
Questo formato di documentazione più esteso è utile quando sono necessarie maggiori spiegazioni sullo scopo di un flusso o quando è necessario descrivere una logica più complessa.
È utile anche quando un flusso fornisce un'API esterna di qualche tipo, fornendo tutti i dettagli necessari per consentire ad altri sviluppatori di utilizzare l'API.
Quando si inizia ad utilizzare Node-RED, probabilmente si comincia ad aggiungere tutti i nodi nello stesso «tab» [scheda] nell'editor. Si potrebbero importare alcuni flussi di esempio condivisi da altri o creare flussi prototipo per testare diverse soluzioni.
Col tempo, questo può portare a un caos di nodi e fili che rende difficile trovare parti specifiche del flusso.
Pensare a come strutturare i flussi all'inizio di qualsiasi progetto di sviluppo può aiutare a mantenerli organizzati e a semplificarne la manutenzione.
Il metodo principale per organizzare i flussi in Node-RED è separarli in più schede all'interno dell'editor. Esistono diverse strategie utilizzabili a questo scopo.
Se si riescono a identificare componenti logici separati dell'applicazione, valutare la possibilità di inserirli in schede separate.
Per un'applicazione di domotica, si potrebbe inserire la logica di flusso per ogni stanza in una scheda separata per riflettere lo spazio fisico. Oppure si potrebbero separare i flussi in base alla funzione, in modo che tutti i flussi relativi all'illuminazione siano in una scheda e quelli relativi al riscaldamento in un'altra.
Se si sta sviluppando un backend API HTTP, ogni scheda potrebbe rappresentare un tipo separato di risorsa a cui l'API accede.
L'obiettivo dovrebbe essere quello di semplificare la «lettura» di un singolo flusso dall'inizio alla fine. Mantenere tutto in un'unica scheda può aiutare a raggiungere questo obiettivo.
Un altro aspetto da considerare è se si sta lavorando con altri sviluppatori sulla stessa applicazione Node-RED. È molto più facile gestire l'unione delle modifiche se si trovano in schede separate. Se si hanno sviluppatori con ruoli o specializzazioni diverse, considerare come ciò potrebbe influire sull'organizzazione dei flussi.

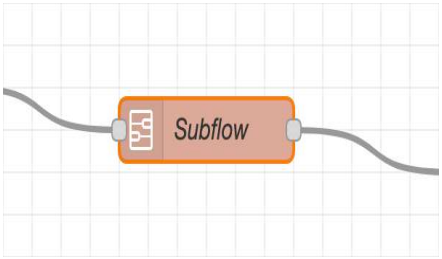
Creazione di Flussi Riutilizzabili

Durante la creazione dei flussi, si potrebbero trovare alcune parti comuni riutilizzabili in più punti. Si dovrebbe evitare di avere più copie di queste parti comuni sparse nei flussi, poiché diventano più difficili da gestire: ci si ritroverebbe con più punti in cui applicare le correzioni e se ne potrebbe facilmente trascurarne una.
Node-RED offre due diversi modi per creare flussi riutilizzabili: «Nodi Link» e «Subflow».



Link nodes

I **Nodi Link** consentono di creare un flusso che può passare da una scheda all'altra nell'editor: aggiungono un collegamento virtuale dalla fine di un flusso all'inizio di un altro.

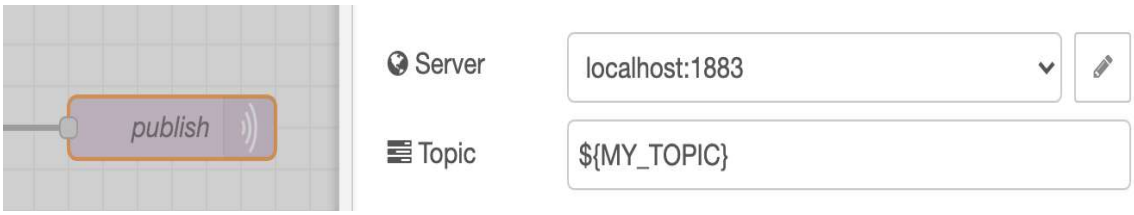


Subflows

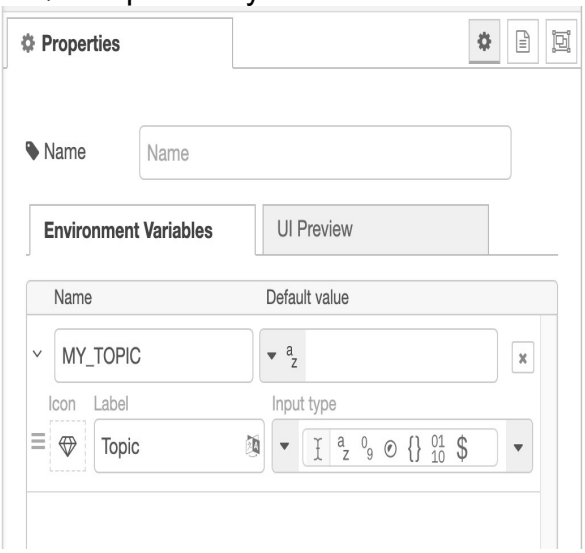
I **Subflow** consentono di creare un nuovo nodo nella palette la cui implementazione interna è descritta come flusso. È quindi possibile aggiungere nuove istanze del sottoflusso ovunque si aggiungerebbe un nodo normale.
Esistono alcune importanti differenze tra i due approcci. I «Nodi Link» non possono essere utilizzati nel mezzo di un flusso, dove i messaggi vengono trasmessi tramite il link e poi restituiti al completamento dell'altro flusso. Possono essere utilizzati solo per avviare o terminare un flusso. Possono anche essere collegati a più di un altro «Link Node». Ciò consente di inoltrare messaggi a più flussi o di far sì che più flussi inoltrino messaggi in un unico flusso. Possono essere utilizzati all'interno di una singola scheda per aiutare i flussi a scorrere nell'area di lavoro senza troppi collegamenti che si incrociano da destra a sinistra.
I «Subflow» appaiono come nodi normali, quindi possono essere utilizzati in qualsiasi punto di un flusso. Tuttavia, ogni istanza del sottoflusso è indipendente dalle altre. Qualsiasi contesto di flusso all'interno del «Subflow» sarà limitato alle singole istanze. Se il «Subflow» crea una connessione a un sistema remoto, ogni istanza creerà la propria connessione.

Personalizzazione dei «Subflow»

Quando si creano «Subflow», potrebbe essere necessario personalizzarne il comportamento in qualche modo. Ad esempio, modificando l'argomento MQTT su cui pubblica.
Un modo per farlo è impostare `msg.topic` su ogni messaggio passato al `Subflow`. Ma questo richiede l'aggiunta di un nodo «Change» davanti a ogni istanza del `Subflow` per impostare il valore desiderato.
Un modo più semplice per farlo è utilizzare le proprietà del `Subflow`. Si tratta di proprietà che possono essere impostate sull'istanza del `Subflow` e che appaiono come variabili di ambiente al suo interno.
Nell'esempio MQTT, si potrebbe prima configurare il nodo per la pubblicazione su `${MY_TOPIC}`.



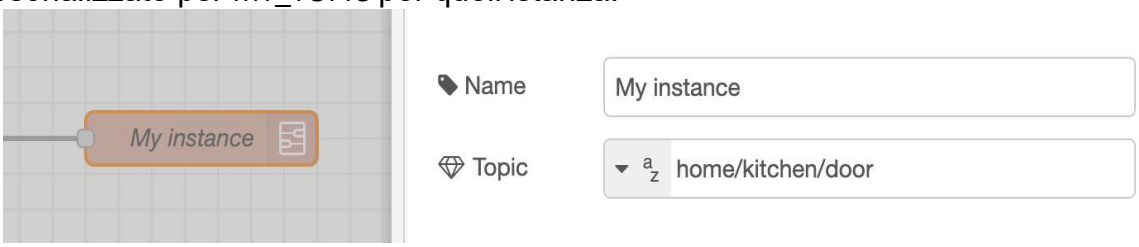
MQTT topic set by environment variables



Adding a subflow property

Quindi aggiungere MY_TOPIC come proprietà del Subflow.

Quando un utente modifica una singola istanza, può quindi fornire un valore personalizzato per MY_TOPIC per quell'istanza.



Customising a subflow instance property

Questo pattern può essere applicato a qualsiasi campo di configurazione del nodo che consenta di immettere il valore direttamente. Al momento non funziona per i campi esposti come «checkbox» o altri elementi personalizzati dell'Interfaccia Utente.

Gestione delle Informazioni sullo Stato

Un'altra considerazione da fare è come gestire le informazioni sullo stato nei flussi. Ad esempio, tenere un conteggio del numero di messaggi che attraversano un flusso o lo stato corrente di un sensore esterno.
Node-RED fornisce il sistema Context per la gestione dello stato all'interno del runtime. Il contesto può essere limitato alla stessa scheda, Subflow o reso disponibile a livello globale.
Se un'informazione sullo stato è necessaria solo per i nodi di una scheda specifica, è consigliabile utilizzare lo «scope» del flusso anziché quello globale. È inoltre consigliabile scegliere con cura i nomi delle variabili di contesto: assicurarsi che siano descrittivi e facili da identificare.

Un'altra opzione è quella di gestire lo stato al di fuori di Node-RED, ad esempio utilizzando messaggi MQTT conservati o un database di qualche tipo. Queste opzioni aggiungono una dipendenza esterna da gestire e non sono integrate in modo così pratico come Context, ma possono anche essere utilizzate insieme a Context e non come una sostituzione completa. Ad esempio, per condividere le informazioni sullo stato tra più istanze di Node-RED o, nel caso di MQTT, essere in grado di attivare un flusso ogni volta che un valore cambia.

Personalizzazione dei Flussi per Diverse Piattaforme

Le variabili d'ambiente possono essere utilizzate più ampiamente all'interno di Node-RED per creare flussi personalizzabili per diverse piattaforme senza dover apportare modifiche manuali.
Ad esempio, si potrebbe avere un flusso che si prevede di eseguire su più dispositivi, ma ogni dispositivo dovrebbe sottoscrivere il proprio argomento MQTT univoco.
Come nell'esempio del Subflow precedente, è possibile configurare il nodo MQTT per pubblicare su ${MY_TOPIC} e quindi impostarlo come variabile d'ambiente prima di eseguire Node-RED. Ciò consente di gestire le personalizzazioni specifiche del dispositivo separatamente dai flussi che dovrebbero essere comuni a tutti i dispositivi.
Lo stesso approccio può essere utilizzato quando i flussi potrebbero essere eseguiti su sistemi operativi diversi, dove il percorso a un file utilizzato dai flussi potrebbe essere diverso a seconda del sistema operativo.
I nodi Inject e Change possono accedere alle variabili d'ambiente utilizzando l'opzione «env» nel loro TypedInput. Il nodo Function può utilizzare la funzione `env.get()`.

Gestione degli Errori

Node-RED fornisce i nodi Catch e Status come strumenti per creare flussi in grado di rispondere agli errori. Per ulteriori informazioni sul loro utilizzo, consultare la guida utente.
Poiché non esiste un'associazione visiva diretta tra un nodo Catch e i nodi a cui si riferisce, è opportuno valutare come posizionarli per mantenere i flussi leggibili.
Posizionarli vicino alle parti del flusso a cui corrispondono può essere utile, ma è necessario fare attenzione a non sovraffollare i flussi.
Un altro approccio consiste nel raggruppare tutti i flussi di gestione degli errori sotto il flusso principale, rendendo il path "corretto" chiaramente distinto dai path di errore.
Anche assegnare un nome chiaro ai nodi Catch è molto importante per identificare facilmente gli scenari che devono gestire.
Qualunque approccio si scelga, è importante cercare di essere coerenti tra i diversi flussi.
Node-RED consente di iniziare rapidamente a sviluppare applicazioni trascinando i nodi e collegandoli tra loro per creare flussi. Questo può essere un ottimo modo per iniziare, ma con la crescita dei flussi nel tempo, può portare ad applicazioni più difficili da gestire.
Questa guida fornisce alcuni consigli e «best practice» su come creare flussi Node-RED riutilizzabili, più facili da gestire e più robusti.
Questa guida presuppone che si abbia già familiarità con l'utilizzo di base di Node-RED.
Per maggiori informazioni sull'utilizzo di Node-RED, la Guida Utente [https://nodered.org/docs/user-guide/] e il Cookbook [https://cookbook.nodered.org/] sono ottime risorse per iniziare.

Struttura dei Flussi

Questa sezione illustra come organizzare i flussi, le strategie per suddividerli in componenti più piccoli e riutilizzabili e come personalizzarli per diverse piattaforme.

Progettazione dei Messaggi

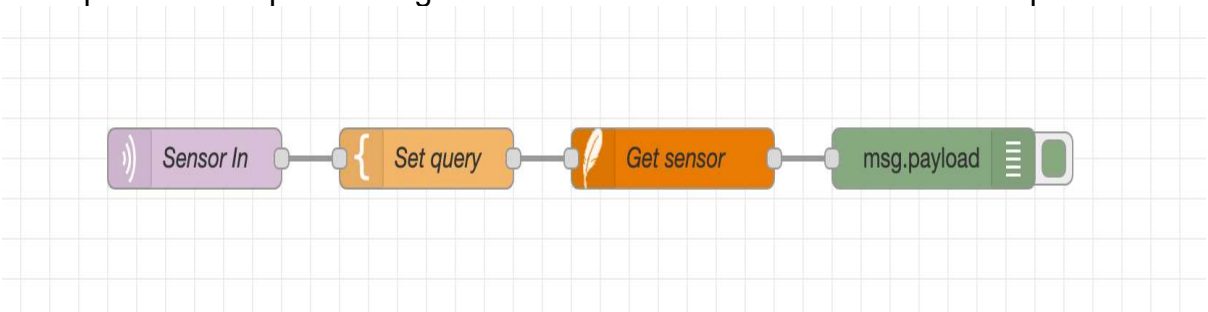
Questa sezione esamina come la progettazione dei messaggi possa contribuire a creare nodi e flussi che funzionino bene insieme e siano più facili da gestire.

Documentazione dei Flussi

Tutto il buon codice dovrebbe essere corredato da una buona documentazione. Questa sezione esamina gli strumenti e le tecniche che Node-RED fornisce per aiutare a documentarli.
I messaggi che attraversano un flusso sono semplici oggetti JavaScript a cui è possibile impostare delle proprietà.
Di solito hanno una proprietà `payload`, che è la proprietà di default con cui la maggior parte dei nodi funzionerà.
Per ulteriori informazioni sui messaggi in Node-RED, consultare la sezione Lavorare con i Messaggi della guida utente.
Questa sezione esamina alcune delle scelte da effettuare quando si decide come strutturare i messaggi nei flussi.

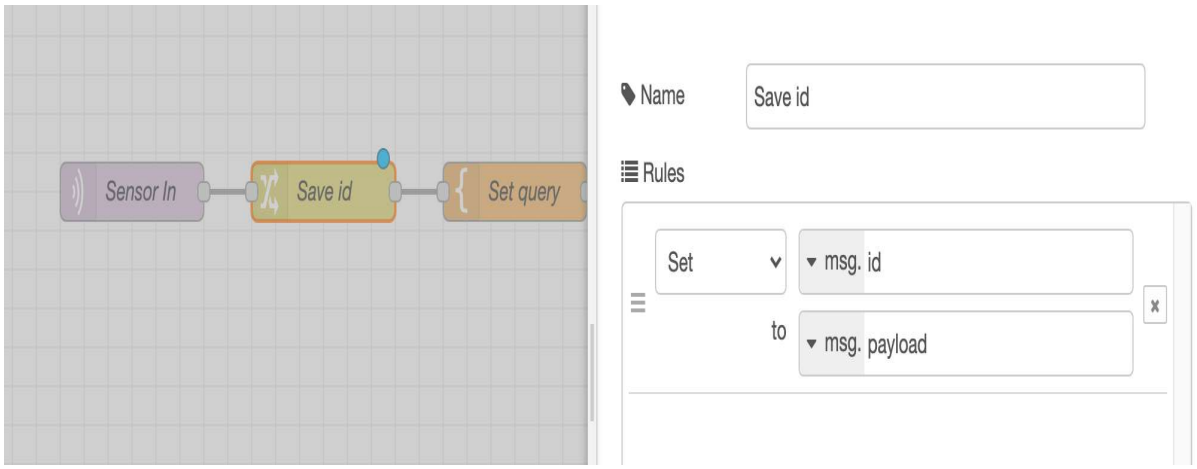
Lavorare con msg.payload

Quando si creano flussi, la scelta delle proprietà utilizzate per un messaggio sarà determinata in gran parte dalle esigenze dei nodi nel flusso.
La maggior parte dei nodi si aspetta di lavorare con `msg.payload` e questo guiderà la maggior parte delle scelte effettuate.
Ad esempio, si consideri un flusso che riceve un ID nel payload di un messaggio MQTT. Utilizza quindi tale ID per interrogare un database e trovare un record corrispondente.



MQTT to database query

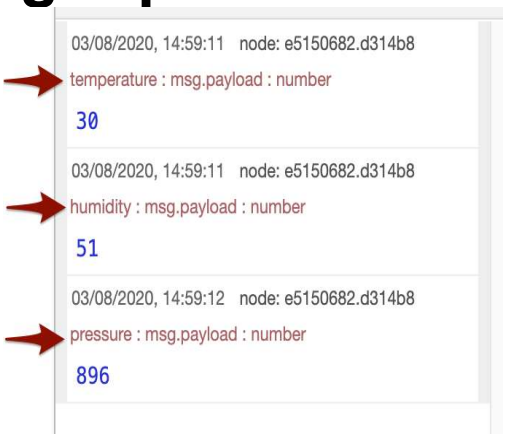
Il nodo del database inserirà il risultato nel payload del messaggio inviato, sovrascrivendo il valore ID originale. Se il flusso deve poter fare riferimento a quel valore ID in un secondo momento, può utilizzare un nodo Change per copiare il valore in un'altra proprietà che non verrà sovrascritta.



Using a Change node to copy the payload to msg.id

Questo evidenzia un principio importante: i nodi non devono modificare o rimuovere proprietà sui messaggi che non sono correlate alla loro funzionalità.
Ad esempio, nella maggior parte dei casi, un nodo Function dovrebbe inviare sullo stesso oggetto messaggio ricevuto anziché creare un nuovo oggetto messaggio.

Utilizzo di msg.topic



msg.topic shown in debug sidebar message

Diversi nodi trattano anche msg.topic come se avesse un significato speciale. Potrebbe essere utilizzato per identificare l'origine del messaggio o per identificare diversi "stream" di messaggi sugli stessi flussi. Viene inoltre visualizzato nella barra laterale Debug con ogni messaggio.
Ad esempio, il nodo «MQTT In» imposterà msg.topic sul «topic» [argomento] su cui è stato ricevuto il messaggio. Il nodo Delay può quindi essere configurato per limitare la frequenza dei messaggi in base al loro argomento.
Sebbene il flusso non utilizzi nodi che dipendono direttamente da msg.topic, può essere utilizzato per fornire informazioni contestuali aggiuntive su un messaggio. Tuttavia, è necessario prestare attenzione se in seguito si introducono nel flusso nodi che dipendono dal suo valore.

Progettazione delle Proprietà del Messaggio

Quando si progetta un nodo o un Subflow per il riutilizzo, le proprietà del messaggio con cui funziona e le proprietà che imposta fanno tutte parte dell'API che espone. Come tutte le API, deve essere progettata con cura e attenzione. Questo vale anche per i flussi.
Un approccio consiste nel collocare tutto nel payload. Ad esempio:

```
{
 "payload":
{
"temperature":123,
"humidity":50,
"pressure":900
}
}
```

Questo può essere comodo per mantenere i dati insieme, ma può anche portare a molti spostamenti di proprietà, poiché i nodi successivi si aspettano di operare su msg.payload e non su una proprietà sottostante.
Un approccio diverso, come quello del nodo Twitter, consiste nell'inserire le informazioni più "interessanti" nel payload, in questo caso il testo di un tweet, e inserire i metadati completi forniti dall'API in una proprietà separata msg.tweet.
Non esiste una risposta univoca su come strutturare il messaggio, ma è importante concentrarsi su come il nodo o il flusso verrà utilizzato nei casi più comuni.
Come per la programmazione in generale, anche la scelta di nomi di proprietà efficaci è importante. Dovrebbero essere autodescrittivi per facilitare il debug e la comprensione del flusso in seguito. Ad esempio, msg.temperature è molto più comprensibile di msg.t.
Dovrebbero anche evitare proprietà di uso comune come reset e parts, che hanno un significato speciale per alcuni nodi.