VERBALI ESTERNI - INVERVISTE CON GUIDI:

sottolineato in rosso il succo del discorso e le parole chiave

17 Marzo 2017 - REDIREZIONE DINAMICA - EMBEDDINGDINAMICO - BINDING DINAMICO

NETBREAK:

circa l'APIGateway dobbiamo capire precisamente come:

- includere/creare a tempo di runtime un'interfaccia
- includere/creare a tempo di runtime un'outputport

Supponiamo che il gateway sia 1 servizio. Quando il client usa un certo servizio la richiesta verra' fatta usando l'interfaccia del gateway e quindi al gateway arrivera' una request su una operation definita nell'interfaccia gateway (es.), fa dei controlli usando un altro servizio... e dopo che tutto e' ok include a runtime l'interfaccia del servizio richiesto dal client prendendola da un file o da un database (comunichera' con un servizio apposito)... e definisce una outputport per permettere di comunicare con il servizio in cui verra' inclusa quindi l'interfaccia appena creata, la location e il protocol.

GUIDI:

confermo che potete fare binding a runtime di protocollo e location.
Potete creare outputPort dinamicamente utilizzando **setOutputPort** del servizio Runtime
http://docs.jolie-lang.org/#!documentation/jsl/Runtime.html
Però creerete una outputPort senza interfaccia associata perché al momento non è possibile fare binding dell'interfaccia a runtime
Potete anche impostare dinamicamente una redirection sulla porta appena creata con **setRedirection**
potete embeddare al volo un file jolie con **loadEmbeddedService**

Cosa non potete fare ad oggi:
non potete fare binding a runtime di una interfaccia su una porta (sia di input che di output)

Quindi: se dovete semplicemente impostare una redirection dinamica avete gli strumenti per farlo
Se invece dovete impostare una aggregation dinamica siete costretti a riavviare il server di endpoint tutte le volte che fate una modifica (ameno di non introdurre modifiche al linguaggio, il che è possibile ma bisogna sempre valutarne i tempi)

A mio avviso la soluzione più interessante è proprio quella della redirection dinamica unita ad un embedding dinamico all'interno del quale eseguire il microservizio che riceve effettivamente la chiamata (ed eventualmente la ridirige a sua volta al microservizio target)

Riassumo il problema per vedere se ho capito:

- chiami il gateway per farti dare l'interfaccia con la definizione della operation da chiamare
- ricevuta l'interfaccia ti costruisci al volo un servizio client da embeddare per fare la chiamata al gateway
- il gateway fa redirection verso il servizio da chiamare

E' giusto?

MONTESI:

Jolie non supporta ancora la creazione di interfacce a runtime. Per la creazione di outputports invece, setOutputPort@Runtime crea o modifica un'output port chiamata .name (dal messaggio di request), settandone anche la location con .location ed il protocollo con .protocol.

Per poi creare una redirection basta usare setRedirection@Runtime, nel caso si voglia usare redirections nell'API gateway.

Ad esempio per creare una redirection equivalente a questa:

inputPort MyInput { Redirects: X => Y }

basta fare:

setRedirection@Runtime({ .inputPortName = "MyInput", .resourceName = "X", .outputPortName = "Y" })()

La redirection e' spiegata in questa pagina: http://docs.jolie-lang.org/#!documentation/architectural_composition/redirection.html

questo esempio dovrebbe essere funzionante:

https://github.com/jolie/examples/tree/master/04_architectural_composition/06_aggregation/04_redirection/01_static_redirection

Altrimenti, sezione 7.3.1 a pagina 101 della mia tesi di master spiega come fare meta-servizi che usino redirezione in modo dinamico:

http://fabriziomontesi.com/files/m10.pdf

6 Aprile 2017 - PRIMO PROTOTIPO GATEWAY - COME RICHIAMARE SERVIZI DA GATEWAY - - INTERFACE EXTENDER - GATEWAY DISTRIBUITO

NETBREAK:

Salve,

Qui il codice e in allegato un pdf che spiega le cose fatte nel prototipo https://github.com/dans3r/gateway (primo prototipo base gateway non piu' presente nel link)

I miei dubbi:
- gestione errori usando questo modello
- giusto l'uso di uno standard diverso per invocare i ms?
- usando i redirect si evita l'uso di uno standard diverso? in effetti adesso mi guardero' l'esempio funzionante nel link dell'altra mail

Grazie e a presto.

GUIDI:

dunque, ho provato a scaricarmi il codice e a me funziona, non ho errori.

Ho solo una perplessità:
impostato così, il client deve sempre chiamare la execute e specificare quale operation vuole eseguire che non è molto"attraente" come feature per uno sviluppatore. Lo sviluppatore vorrebbe chiamare direttamente la API che desidera (aggiungendo i token di autenticazione).
Per questo motivo troverei più idonea la scelta di una

aggregation + courier sul gateway

Oppure avete necessita che il client chiami proprio la execute?

NETBREAK:

Quindi:

Si potrebbe teoricamente fare una richiesta attraverso il gateway usando appunto questo metodo: "interface extender is the keyword used in Jolie for extending operations by overloading their types. The overloaded types contain additional fields exploited within the courier session to perform checks and, before forwarding, they are automatically removed from the message. The interface extender syntax follows." In tal modo il client fa le richieste con la stessa interfaccia del servizio senza inventarsi protocolli ad hoc potendo overloaddare ogni volta i tipi delle richieste per fare check.

```
inputPort AggregatorPort {
 // Location definition
 // Protocol definition
 Aggregates:
  outputPort_1 with extender_id1,
  //  ...
  outputPort_n with extender_idn
}
```

avendo quindi questa inputport si usa l'aggregation per dire che l'interfaccia del servizio a cui verra' inoltrata la richiesta sara' estesa dal richiedente e prima della redirection scarta le info extra

GUIDI:

Si può utilizzare la redirection per unificare l'endpoint di accesso al gateway.
La redirection non fa check sull'interfaccia e inoltra il messaggio ad un servizio con aggregation che applica la courier che a sua volta reinoltra al target finale.
In allegato la possibile architettura: ...

immaginavo ci fosse una specie di catalogo risorse da cui prendere le API.
A quel punto dal catalogo mi scarico l'endpoint con l'URL da passare al redirector.

Il redirector, verrà notificato tutte le volte che c'è un nuovo servizio con il suo endpoint
ed a quel punto lui si carica al volo la outputPort sulla quale fare redirezione

nel momento in cui viene creato un servizio nel catalogo e lo si abilita all'utilizzo del GW. verrà mandato un messaggio al redirector per la creazione della outputPort e del servizio ausiliario con aggregatore, quindi in realtà il redirector ha già le informazioni che gli servono.

NETBREAK:

quindi lei dice: ogni volta che avvio il redirector lui si inizializza istanziando tutte le outputport e tutti i redirect ai servizi ausiliari... e pero' in aggiunta aggiungo un operazione che viene richiamata dal microservizio gestore dei servizi che ogni volta che un venditore aggiunge un servizio crea un'outputport dinamicamente e un ulteriore servizio ausiliario.

GUIDI:

esatto, si.

NETBREAK:

bisogna pero' tenere conto che tutti i servizi ausiliari saranno pero' da embeddare nel gateway. Quindi possibilmente posso avere un gateway che ri.chiede 100000000 servizi dove in un unico servizio ho embedded 100000000 servizi ausiliari

GUIDI:

puoi sia definire delle operation sue che aggiungere la redirection. Per l'embedding dipende non è strettamente necessario nel senso che i servizi ausiliari potrebbero essere anche indipendenti quindi se si ragiona bene sulle politiche di scaling si potrebbe iniziare embeddando, poi superato un certo

limite potrebbero essere deployati in modo indipendente ecc... è un'architettura estremamente flessibile. Si potrebbero avere anche più punti di accesso per i client senza necessariamente averne uno solo (ossia avere più redirector). Le policy del gateway in questo caso sono distribuite all'interno delle courier dei servizi ausiliari (bisognerà trovare un nome per questi servizi, non possiamo chiamarli ausiliari)

NETBREAK:

quindi lei dice che in questo modo potrebbe essere reso tutto distribuito partendo da un modello inizialmente centralizzato con pochissime modifiche? in questo senso semplicemente associare location e protocollo diversi a ogni set di servizi? (in soldoni)

GUIDI:

sì tanto avere un servizio ausiliario embeddato oppure no non fa alcuna differenza (cambia solo la location della outputPort nel redirector). Cambia solo il fatto che in un caso embeddate e nell'altro dovete lanciare il servizio nella macchina di destinazione.

10 Aprile 2017 - SECONDO PROTOTIPO GATEWAY - AUTOMATIZZAZIONE COURIER - INSTALLAZIONE JOLIE SU SERVER

NETBREAK:

Questo e' il secondo prototipo del gateway, fatto in base a quello che e' stato detto.
https://github.com/dans3r/gateway
gateway in jolie. Per testarlo bisogna avviare redirector.ol, helloservice.ol, client2.ol. Prossimamente facciamo:

1. Quando il publisher aggiunge un'interfaccia un 'compiler' genera l'interfaccia estesa da fornire al client
2. Il redirector prende un modello con tutti i servizi con i giusti dati e genera tutti i file con il courier per tutti i servizi
3. Aggiungiamo error handling generale dentro i courier, i check delle key e tutto quel che serve, uguale per tutti i servizi
4. Possibilmente per fare un gateway distribuito su n = infinito server facciamo una procedura usando il pattern strategy che generi redirector un subset di servizi, sulla base di un tag oppure altri parametri. Noi faremo inizialmente un redirector centralizzato, ma estendendo questa procedura lo si potra' fare distribuito appunto su quanti server si vorra'.

N.B. al client forniamo l'interfaccia estesa gia' con l'outputport giusta istanziata, quindi il client si dovra' preoccupare solo di fare le richieste senza dover configurare. In tale modo la user xperience sara' massimizzata perche' un client dovra' solo sapere Jolie, senza dover imparare uno standard interno nuovo.

I courier: come puo' notare i courier differiscono tra loro solo per le interfacce e le location e possibilmente i protocolli

GUIDI:

direi che ci siamo

NETBREAK:

come si installa un servizio Jolie su un server?

GUIDI:

ci sono due possibilità al momento. In server dove è già preinstallato jolie, basta copiare la cartella e poi lanciare il processo jolie che lancia il main. Oppure utilizzare docker
(però è più sperimentale come cosa anche se forse a voi potrebbe risultare più interessante.
Oppure qui:

http://docs.jolie-lang.org/#!documentation/containers/docker.html
trovate informazioni. C'è poi un progetto sperimentale iniziato da un tirocinante per creare un servizio jolie wrapper di docker così da poterlo invocare direttamente con jolie, ma è sperimentale quindi potreste incontrare cose che non funzionano
https://github.com/jolie/jocker.

13 Aprile 2017 - 1 SERVIZIO = n SOTTOSERVIZI, per ogni SOTTOSERVIZIO k INTERFACCE - COME PUBLISHER INSERISCE DOCUMENTAZIONE - IMPORTANZA DATI MONITORING - SITO MASHAPE OTTIMO PER FRONTEND

NETBREAK:

Stavo pensando al concetto di servizio in Jolie. Finora e' stato assunto per semplificare il problema che un servizio avesse una sola interfaccia. Pero' forse un fornitore di servizi potenzialmente potrebbe aver bisogno di fornire piu' interfacce per lo stesso servizio.

In questo momento stiamo codificando delle parti delicate del sistema, parti che coinvolgono l'upload dei servizi e le parti che servono per completare il funzionamento del gateway, che presto conto avra' almeno le funzionalita' di automazione minine . Ti chiedo quindi. E' giusto assumere 1 servizio = 1 interfaccia oppure sarebbe bene prevedere 1 servizio = n interfacce? Mi sono infatti accorto che il modello del database attuale prevede 1 servizio = 1 interfaccia e sarebbe bene in caso correggere questo.

GUIDI:

Un servizio = n interfacce

Ma anche un servizio = n inputPort (se necessario)

NETBREAK:

L'altra cosa riguarda il come un publisher dovrebbe inserire la documentazione. Sarebbe utile avere un modo standard da imporre al publisher per avere documentazioni dei servizi in una forma sempre uguale. Al momento i nostri casi d'uso prevedono l'inserimento della documentazione come pdf oppure come html.

Cosa che vuol dire che ogni potenziale publisher ipotetico fara' un po' come gli pare e questo penso diminuisce la usabilita' dell'app web a causa di un aumento della complessita' richiesta a un user umano (uso in piu' di energie ogni volta per capire dove guardare nella documentazione), appunto perche' ogni documentazione sara' scritta in modo diverso probabilmente.

Quello che avevo pensato era questo: aumentare leggermente la complessita' computazionale richiesta al publisher all'atto di inserire la documentazione al fine di dimuire di molto la complessita' computazionale richiesta al client che consulta.

In pratica pensavo di usare questo: https://quilljs.com

e' un editor open source "what you see is what you get", che io vorrei usassimo nella sua versione base (senza prevedere uso immagini, video, file ecc...) per permettere ai publisher di scrivere su un editor interno 'alla word' una documentazione fatta bene con elenchi puntati, formule ecc... a questo vorrei possibilmente applicare un filtro auto-complete, che permetta di facilitare la scrittura delle interfacce/documentazione. Per interfacce qui non intendo l'interfaccia .iol. Insomma per avere una documentazione tutta standardizzata un po' come quella dei servizi del sito di jolie-lang.

I miei colleghi mi hanno fatto notare che tu avevi gia' proposto l'utilizzo di https://swaggerhub.com/api-documentation/

Nel caso in cui ci sara' poco tempo da dedicare a questo pezzo del sistema, caso peggiore, potremmo essere comunque costretti a continuare sulla strada dei gia' presenti use case, coi pdf/html ma in caso il tempo ci sia, vorrei fare la scelta piu' adeguata.

GUIDI:

C'è un tool che si chiama joliedoc che viene installato con jolie e che genera la documentazione a partire da una porta. Se commenti opportunamente l'interfaccia in automatico i commenti vengono riportati in documentazione. Se volete potete pensare anche di estendere il tool se ritenete che possano essere introdotte nuove feature (la cosa andrebbe però discussa un po' con il resto della community).

Non conosco quilljs, quando riesco ci guardo. Nulla in contrario cmq se lo volete utilizzare.

La documentazione nel sito di Jolie e' generata proprio con joliedoc.
Ma tu intendi che il developer sviluppa l'interfaccia direttamente sul web oppure ne importa una e poi completa la documentazione sul web?

Per quanto riguarda l'utilizzo di https://swaggerhub.com/api-documentation/ io ho già fatto un traduttore da interfacce jolie ad interfacce JSON swagger. Però questo formato va bene se si vogliono pubblicare api JSON di tipo rest. In realtà c'è anche già il router rest in questo caso fatto proprio con jolie. Fatemi sapere se lo volete inserire perché vi spiego come usarlo.

NETBREAK:

Al momento non e' definito come viene aggiunto codice html, penso se si sceglie questa strada (nel caso peggiore) e' meglio l'inserimento diretto come su ebay.

Adesso stiamo scrivendo codice del servizio che gestisce i servizi e una procedura che all'atto dell'inserimento di un servizio, a partire dall'interfaccia fornita 1. ne verifica la correttezza sintattica 2. ne estrapola (nome, location, protocol) 3. genera l'interfaccia da fornire al cliente includendo la giusta outputport. .

A proposito delle operazioni da effettuare sull'interfaccia all'atto dell'inserimento da parte di un publisher: e' necessario che facciamo in modo di generare un tester.ol, che ogni volta prova tutte le operations del servizio per verificarne il funzionamento oppure non necessariamente tutte le operations ma anche una sola per provare almeno che li' effettivamente c'e' un servizio? Oppure possiamo lasciare quest'ultima cosa come facoltativa?

GUIDI:

Riguardo ai test 'Domandona'. Ci possono essere diverse strade se però nel vostro modello voi non siete responsabili del deployment ma siete solo collettori allora forse si può evitare il test. Sarebbe invece più importante collezionare i dati di monitoring.

NETBREAK:

Ti volevo chiedere se esiste un esempio da qualche parte per quanto riguarda il passaggio delle immagini e relativo controllo dei formati con jolie. Mi interessava capire il modo migliore di farlo in Jolie. Avevo guardato il servizio "File" che penso sia da usare

GUIDI:

intendi immagini inviate via web?
In quel caso vi conviene utilizzare leonardo ma forse è meglio se organizziamo una mini skype conf che facciamo prima. si puo' si usare 'File'.

19 Aprile 2017 - ESEMPIO UPLOAD IMMAGINI - COME RISOLVERE CONFLITTI SU INTERFACCE CLIENT - COMUNICAZIONE DA ANGULARJS

GUIDI:

puoi fare una form in cui carichi il file e come action indichi una operation che implementi sul leonardo con il quale la webapp parla.
A quel punto lato server ti ritrovi il file che puoi salvare usando una writeFile con format="binary".

ecco un esempio per upload di immagini in uploadimg.zip:

lanci leonardo.ol, poi vai sul browser http://localhost:8000/index.html, selezioni un pdf e spingi upload. Nella cartella di leonardo ti ritrovi un file prova.pdf

NETBREAK:

Cosa ne pensi di https://market.mashape.com/explore?sort=developers come client web. Non lo abbiamo fatto noi questo, pensavo pero' che fosse carino. Quindi se per te va bene possiamo "prendere ispirazione".

GUIDI:

direi che è un ottimo spunto di ispirazione

NETBREAK:

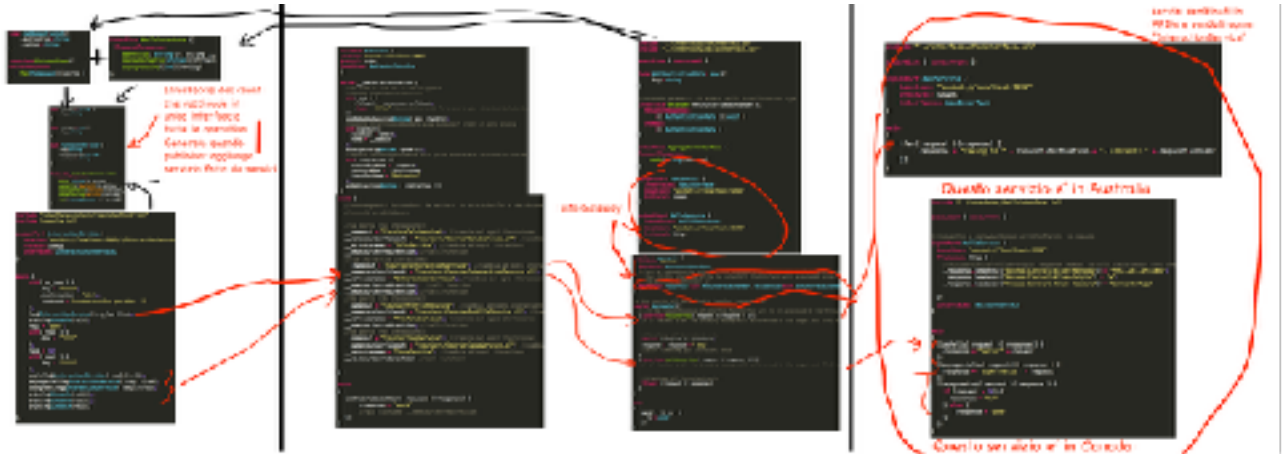
Stavo pensando: Se un publisher pubblica un servizio fatto di 3 servizi (ognuno con propria location e protocol) e il primo servizio ha un'operation con nome uguale a un'operation del terzo servizio, se io creo un'interfaccia unica (per il client) aggregando tutte le operation dei vari servizi, non vi sono poi conflitti potenzialmente?

qui esempi di come ho immaginato soluzioni ai problemi:

https://github.com/dans3r/gateway

vedi "interactioninterfaceC.iol" e “calc_interfaceC.iol"

ho provato sia a legare al gateway un servizio con tante interfacce ma tutto su stessa location, sia un servizio composto da altri servizi (con varie interfacce) con proprie location.



Nell’immagine flow di servizio composto da servizi indipendenti (tante location)

GUIDI:

Ok quindi se ho capito bene siamo nel caso in cui il publisher decide di aggregare fax ed helloservice DENTRO al gateway, programmando la cosa da interfaccia web. Il gateway perciò vede i due servizi singoli mentre il client vede un unico servizio aggregato.

Il gateway ha quindi una outputPort per ogni insieme di interfacce con location diversa.

Mitico.

Direi che va bene.
A questo punto c'e' un possibile conflitto dei nomi delle operations nel momento in cui il publisher decide di aggregare sul gateway i due servizi. Lì se c'è un conflitto di nomi devi avvisarlo con un warning dandogli due possibilità:
- correzione automatica: introduci nell'aggregator una nuova operation che rinomina l'operation che va in conflitto ed al suo interno chiami la operation vecchia.
- correzione manuale: ci pensa lui a risolvere il conflitto sui suoi servizi

Per il client è tutto trasparente ed è giusto mandargli l'aggregazione dell'interfaccia.

C'è un tool che viene dato con jolie che in automatico ti ritorna l'interfaccia totale visibile su una outputPort: jolie2surface (http://docs.jolie-lang.org/#!documentation/other_tools/jolie2surface.html)
Abbiamo denominato questo interfaccia "totale" con il nome di surface.
Forse può esservi utile utilizzarlo.

Puoi anche usare la stessa funzionalità da dentro jolie (senza laciare un comando esterno) utilizzando insieme il javaservice MetaJolie ed il javaservice Parser.
Con il primo estrai i metadati di una porta di un servizio (decodificati in un value jolie) mentre con il secondo c'è una API che ti estrae l'interfaccia a partire dai metadati.

Da non dimenticare la possibilta' delle redirection su inputport nel servizio pubblicato

NETBREAK:

Come vedi sto cercando solo di ridurre eventualmente l'apprendimento d'utilizzo lato client usando un design pattern Facade, fornendo un unica interfaccia. Magari poi il servizio e' composto da 1000 altri servizi distribuiti in tutto il mondo, ma al client che ne usa le operations non deve interessare questo. Penso sia "the whole point of a microservice architecture"... serve solo per facilitare la manutenzione lato developer, non creare dicotomie agli user.

Ti voglio chiedere l'ultimissima cosa d'importanza pero' cruciale.

Tu hai diverse volte citato leonardo (anche per l'esempio dell'upload immagine). Io ho guardato quindi leonardo e ho visto che e' un servizio "da completare" che permette di gestire un sito web con backend jolie.

Assumo quindi che, nonostante tutto, tu preferiresti usassimo leonardo.

Noi abbiamo invece pensato anche quando abbiamo definito l'architettura di fare backend jolie completamente, ma comunicazione "da html" con backend da angularjs usando http. Questa cosa per te non va bene?

se ti ricordi ti avevo fatto vedere un video a tal proposito https://www.youtube.com/watch?v=4M0hRlC2Y10

GUIDI:

Puoi scorporare le api che vuoi esporre su leonardo in un secondo servizio che embeddi e che aggreghi dentro la porta http leonardo (noi lo chiamiamo frontend) così hai le funzionalità separate. Non voglio obbligarvi ad usare leonardo, è chiaro che mi fa piacere, però va benissimo anche che utilizziate un'altra tecnologia.
Angularjs è una tecnologia lato client, nel momento in cui mandi chiamate http al server dietro può starci benissimo anche leonardo senza problemi (però, ripeto, utilizzate la tecnologia che più

gradite). Rimango dell'opinione che con jolie fate abbastanza veloce (anche se non so dirti in termini di tempo quanto perdi e quanto guadagni).
Io ovviamente sono per totalizzare l'uso di jolie in qualsiasi sua forma, ma io, essendo uno dei creatori, non faccio testo quindi vado considerato come un fanatico da questo punto di vista. In generale comunque, imparare jolie aiuta molto a capire i meccanismi della progettazione a servizi (indipendentemente dal fatto che poi verrà utilizzato o meno), quindi cmq sia imparare jolie può essere utile semplicemente per adottare il giusto mindset nell'affrontare la progettazione a servizi. Da questo punto di vista lo sforzo di imparare un linguaggio nuovo verrebbe ricambiato con l'acquisizione di un approccio mentale... ma siete liberi di fare le scelte tecnologiche che piu' gradite.

21 Aprile 2017 - METAJOLIE e PARSER PER INTERFACCE CLIENT USANDO I COURIER, UTILI PER GENERAZIONE AUTOMATICA INTERFACCE

NETBREAK:

Provando jolie2surface ci siamo accorti che 1. fa il check della correttezza delle interfacce ritornando un errore in caso contrario 2. mi controlla se ci sono tipi con stesso nome ma definiti diversamente ritornando un errore 3. putroppo non mi controlla nel caso operations abbiano stesso nome quindi bisogna farlo.

PROBLEMA: funziona solo se c'e' una inputPort associata (o outputport)?

Se io voglio creare una surface a partire dalla interfacce e basta non si puo'?

A tal proposito in che modo sono utili metajolie e parser?

stavo provando anche joliedocs per la documentazione automatica.

Possibilmente vorrei poi generare la documentazione automaticamente dall'interfaccia Client e salvarla in un modello sql e l'idea che ho al momento e' fornire ai publisher un modo da interfaccia web di completare i docs solo con una descrizione aggiuntiva per ogni metodo (cosa fa), con o senza codice esempio.

GUIDI:

Per esempio se chiami la getInterface passandogli la descrizione di un'interfaccia che hai tra i metadati, restituisce l'interfaccia scritta in jolie.

La cosa interessante è che i metadati sia di una input che di una outputPort sono definiti tramite un tipo Participant che puoi dare in pasto alla getSurface del parser per ottenere la surface di quella porta (in questo caso anche una outputPort).

salva questo codice in un file chiamato example.ol e lancialo. Ti stampa la sua stessa interfaccia. L'unico problema è che al momento te la stampa senza tipi. I tipi sono comunque contenuti nei metadati ma devi essere tu che te li scorri tutti e capisci quelli che ti servono (è ancora un po' complicato come meccanismo).

Altrimenti sotto utilizzo la getInputPortMetaData per farmi dare la descrizione completa delle inputPort e poi chiedo effettivamente la surface completa di tipi di quella. Secondo me quest'ultima chiamata è quella che fa al caso tuo e che dovrai applicare al servizio courier che aggrega di fatto i servizi. In automatico ti dovrebbe estrarre quello che ti serve comprensivo di outputPort per il client. Se non vuoi la op devi usare la getSurfaceWithoutOutputPort.

```
include "console.iol"
include "string_utils.iol"
include "metajolie.iol"
include "metaparser.iol"
type TestRequest: void {
   .messaggio: string
}
```

```
interface ExampleInterface {
 RequestResponse:
  test( TestRequest )( string )
}

outputPort Example2 {
 Location: "socket://localhost:20001"
 Protocol: sodep
 Interfaces: ExampleInterface
}

inputPort Example {
 Location: "socket://localhost:20000"
 Protocol: http { .format = "xml" }
 Interfaces: ExampleInterface
}


inputPort ExampleSodep {
 Location: "socket://localhost:20002"
 Protocol: sodep
 Interfaces: ExampleInterface
 Aggregates: Example2
}

main {
 r.name.name = "Example";
 r.name.domain = "quello che vuoi"; //serve solo per dare anche un dominio al nome della porta
 r.filename = "example.ol";
 getMetaData@MetaJolie( r )( m );
 getInterface@Parser( m.interfaces[ 0 ] )( inter );
 println@Console( "Interfaccia:\n" + inter )();
 getInputPortMetaData@MetaJolie( r )( m2 );
 getSurface@Parser( m2.input[ 0 ] )( surface );
 println@Console("Surface:\n" + surface )()
}
```

Poi puoi anche utilizzare direttamente le metainfo per la documentazione senza passare da joliedoc.
I commenti vengono già riportati se non erro.

Per farti comparire un commento devi metterlo tra questi token prima di una operation, di un'interfaccia o di un tipo
/**! ..... */

NETBREAK:

ci siamo piu' o meno:

ho usato il tuo esempio, anzi ho contestualizzato usando uno dei miei servizi esempio:

```
include "console.iol"
include "string_utils.iol"
include "metajolie.iol"
include "metaparser.iol"
include "gateway_flow/interfaces/faxinterface.iol"
include "gateway_flow/interfaces/hellointerface.iol"
```

```
type AuthenticationData: any {
   .key:string
}

//estendo parametro di andata delle interfacce con type
interface extender AuthInterfaceExtender {
 RequestResponse:
    *( AuthenticationData )( void )
 OneWay:
    *( AuthenticationData )
}

interface AggregatorInterface {
RequestResponse:
   mock(string)(string)
}

outputPort FaxService {
 Interfaces: FaxInterface
 Location: "socket://localhost:9202"
 Protocol: sodep
}

outputPort HelloService {
 Interfaces: HelloInterface
 Location: "socket://localhost:9500"
 Protocol: http
}


inputPort InteractionService {
 Location: "socket://localhost:2002/!/InteractionService"
 Protocol: http
 Interfaces: AggregatorInterface
 /* here we aggregates the outputPort Fax extended using the extender AuthInterfaceExtender
    thus all the interfaces of the Fax outputPort (FaxInterface) will be extended using
    the AuthInterfaceExtender */
 Aggregates: FaxService with AuthInterfaceExtender, HelloService with AuthInterfaceExtender
}

main {

 r.name.name = "Aggregator";
 r.name.domain = "quello che vuoi"; //serve solo per dare anche un dominio al nome della porta, lo
uso per i registri
 r.filename = "example.ol";
 getInputPortMetaData@MetaJolie( r )( m2 );
 getSurface@Parser( m2.input[ 0 ] )( surface );
 println@Console("Surface:\n" + surface )()


}
```

e stampa:

```
type Request: void {
 .destination[1,1]:string
 .content[1,1]:string
 .key[1,1]:string
}
```

```
type Request: void {
 .destination[1,1]:string
 .content[1,1]:string
 .key[1,1]:string
}
interface InteractionServiceInterface {
RequestResponse:
 mock( string )( string ),
 fax( Request )( string ),
 sayhello( Request )( string ),
 saysuperhello( string )( string ),
 sayagreeting( int )( string ),
 fax( Request )( string )
}

outputPort InteractionService {
 Protocol:http
 Location:"socket://localhost:2002/!/InteractionService"
 Interfaces:InteractionServiceInterface
}
```

in pratica ho ottenuto l'interfaccia del client.

La cosa migliore da fare direi e' :le interfacce che ricevo le metto in un file come questo (potrei generare il courier che avra' questa struttura) e da li' genero l'interfaccia client. Cosa ne pensi?

GUIDI:

Potete utilizzare direttamente il file con le courier che bisogna comunque generare.

Anzi e' il modo migliore perche' cosi' fa anche il check sintattico.

NETBREAK:

Altra domanda al volo: le interfacce le salvo nel db come contenuto, ad esempio:

```
type op: void {
        .a: int
        .b: int
}

interface DivInterface {
 RequestResponse:
  div( op )( double ),
  divanddouble(op)( double )
}
```

oppure come filename?

GUIDI:

per il momento direi che è indifferente, però in prospettiva il file "potrebbe" essere più utile se si volesse integrare il sistema con un git per esempio per gestire i versionamenti e le modifiche nel tempo. Per adesso va bene cosi'.

24 Aprile 2017 - NON SERVE SICUREZZA SU APIKEY - ONEWAY OPERATIONS

NETBREAK:

Due altre domande al volo:

1. sicurezza: io pensavo di usare algoritmi a chiave pubblica per le apikey. Per garantire protezione da attacchi man in the middle ecc... pensavo agli algorithmi a chiave pubblica perche' sono (correggimi se sbaglio) tra quelli piu' potenti.

Sono usati per le transazioni sicure e infatti si puo' essere sicuri che se A manda una Object a B, solo A puo' aver mandato tale Object perche' solo A conosce la chiave privata. Qualcosa del genere.

2.

```
include "console.iol"

type Request:void {
        .destination:string
        .content:string
}

interface FaxInterface {
        OneWay:
                faxwithnoresponse( Request )
        RequestResponse:
                fax(Request)( string )
}

execution { concurrent }

inputPort FaxService {
        Location: "socket://localhost:9202"
        Protocol: sodep
        Interfaces: FaxInterface
}

main
{
        [fax( request )(response) {
          response = "Faxing to " + request.destination + "\n. Content: " + request.content + "\n ok fax mandato"
        }]
        [faxwithnoresponse( request ) {
                a = 3
        }]

}
```

Il codice di prima non compila... non ho trovato esempi al riguardo nel sito o in git su come implementare le operations OneWay

/Users/danser/Desktop/backend/apimservices/faxservice.ol:18: error: expected]. Found token type LCURLY
        at jolie.lang.parse.AbstractParser.throwException(AbstractParser.java:197)...

Mi da questo errore

A questo punto immagino sia abbastanza banale come problema, considerando il tutto. Forse mi sfugge qualcosa?

GUIDI:

1. Questo vuol dire che tu non mandi una API key ma mandi una coppia, chiave pubblica + segreto (che contiene la api key) ... non vorrei che questa cosa pesasse sulle performance. In genere anche quando richiedi una API key a Google te la mandano in chiaro, quindi a mio avviso per il momento è accettabile che non siano protette.

2. c'è un errore sintattico perché la OW non ha un body di codice come la RR. Quindi si scrive così:

```
[faxwithnoresponse( request )] {
        a = 3
}
```

Piccola nota: l'utilizzo delle parentesi [] significa che stai utilizzando un altro operatore che è la scelta non deterministica, ossia lista di possibili operation sulle quali ricevere un messaggio, chi riceve vince le altre perdono e vengono scartate. In generale puoi avere una scelta non det anche dentro al codice di una operation:

```
fax( request )( response ) {
 [ op1()(){ ... } ]

 [ op2()(){ ... } ]

}
```

significa che dopo aver ricevuto fax si mette in ascolto per ricevere su op1 o op2 (una solo delle due, la prima che arriva).
In generale quando usi la [] direttamente sotto il main ti diventa una lista di tutte le op che il servizio mette a disposizione.
In quest'ottica la OW riceve solo un messaggio quindi la sua sintassi è:

[myow(messaggio] { proseguo con il mio codice }

diversa da

[myrr(request)(response){ corpo tra la request e la response }] { prosegue con altro codice se necessario... raro }