

TP1 - Services Web REST et SOAP avec Talend

TP1 - Services Web REST et SOAP avec Talend



Télécharger PDF



Objectifs du TP

Création et consommation de web services SOAP et REST en utilisant l'outil Talend.

Outils et Versions

- **Talend Open Studio for ESB** Version: 8.0.1
- **MySQL** Version *latest*
- **SOAPUI** Version 5.7.0

Talend ESB

Talend ESB est une solution légère, robuste et modulaire pour la création de services web sécurisés ainsi que pour l'intégration d'applications nouvelles ou existantes. Talend participe au développement des composants ESB à travers la communauté Apache. Il collabore avec un

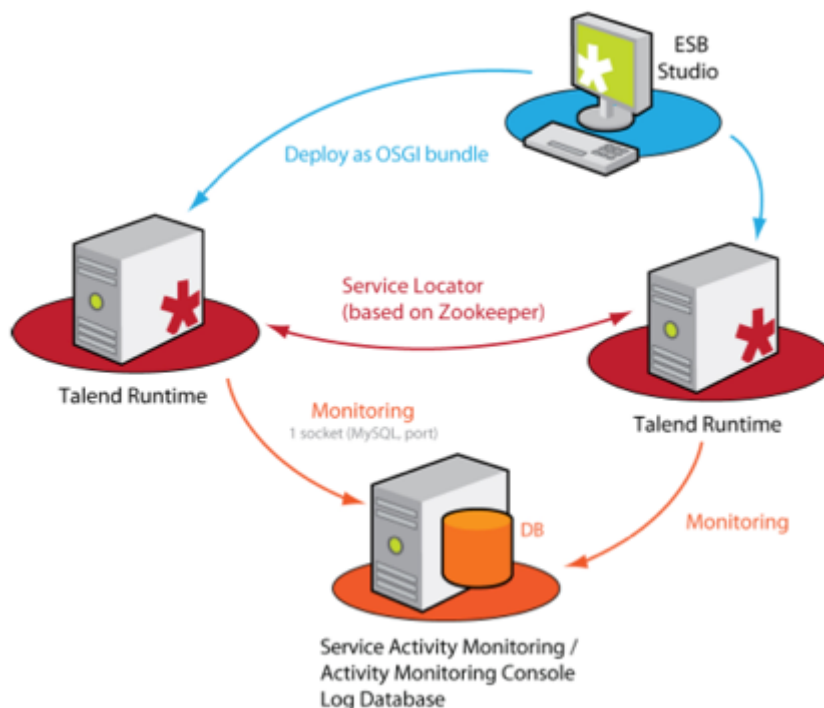
grand nombre de développeurs de la communauté Apache et a fait plusieurs contributions aux projets Apache. Talend fournit:

- Un courtier de messages à haute performance.
- Des options de déploiement flexibles
- Des outils de développement pour Eclipse
- Une interface utilisateur pour l'intégration et la médiation d'applications
- Support pour les services web SOAP et REST
- La médiation et le routage
- Support pour le failover, le monitoring et la sécurité

L'environnement d'exécution standard de Talend ESB est un conteneur OSGi. L'implémentation OSGi fournie avec Talend ESB est Apache Karaf, avec Eclipse Equinox comme environnement d'exécution OSGi. Elle fournit un conteneur léger dans lequel les différents composants et applications peuvent être déployées.

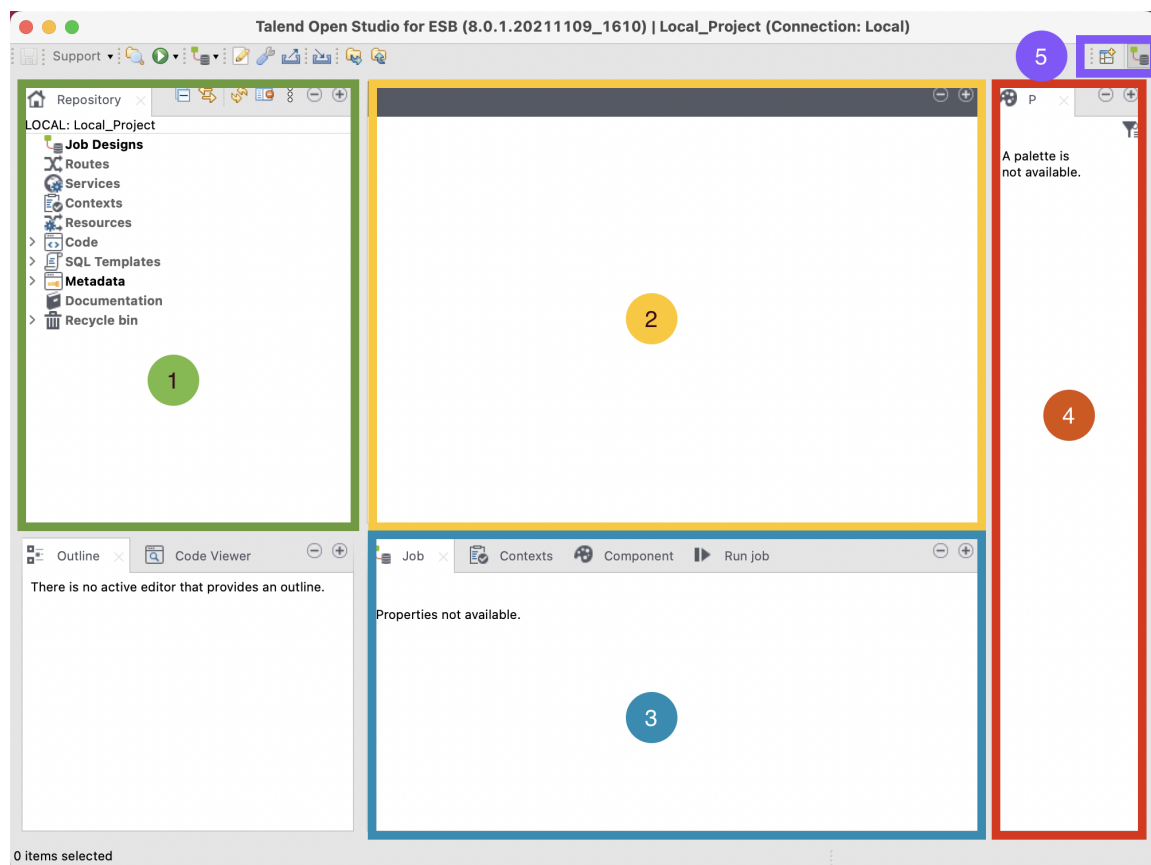
Talend Open Studio for ESB

Talend Open Studio for ESB (TOS-ESB) fournit une interface graphique de développement pour implémenter, compiler, tester et publier des services Web Java, des applications REST, des services de données et des routes de messages. Le déploiement d'applications avec TOS-ESB utilise principalement les trois blocs représentés dans cette figure:






- Le bloc bleu représente l'API Talend Studio, où il est possible d'intégrer des données, des services ou des applications
- Les blocs rouges représentent un ou plusieurs environnements d'exécution Talend déployés dans votre système d'information. Il vous permet de déployer et d'exécuter les Jobs, les routes et les services créés dans Talend Studio. Il est possible d'avoir plusieurs environnements d'exécution, entre lesquels vous pouvez basculer grâce à Talend Service Locator.
- Le bloc orange est une base de données de monitoring, permettant de stocker les informations d'exécution des processus et de l'activité des services.

L'interface utilisateur de TOS-ESB se présente comme suit:



Composant	Fonctionnalité
1	Le traditionnel <i>Repository</i> contenant vos Jobs, services, fichiers, routes...
2	La fenêtre principale, représentant graphiquement la composition de vos jobs et routes

Composant	Fonctionnalité
	La fenêtre contenant les propriétés, la console d'exécution...
	La palette des composants à utiliser
	Les onglets pour le choix de la perspective à utiliser

Service Web SOAP : Helloworld

Commençons par installer Talend ESB en dézipant simplement le fichier téléchargé. Aller ensuite au répertoire Studio, et lancer l'exécutable associé à votre système d'exploitation.

✖ Erreur possible pour les utilisateurs MAC

Pour ceux qui disposent d'un MAC, si l'erreur suivante apparaît: *The TOS_ESB-macosx-cocoa executable launcher was unable to locate its companion shared library.*, réaliser les actions suivantes (ceci se fera une seule fois):

- Ouvrir un terminal sous le répertoire Studio
- Lancer la commande suivante:

```
xattr -c TOS_ESB-macosx-cocoa.app
```

- Lancer ensuite l'exécutable une deuxième fois. Cela devra le débloquent.

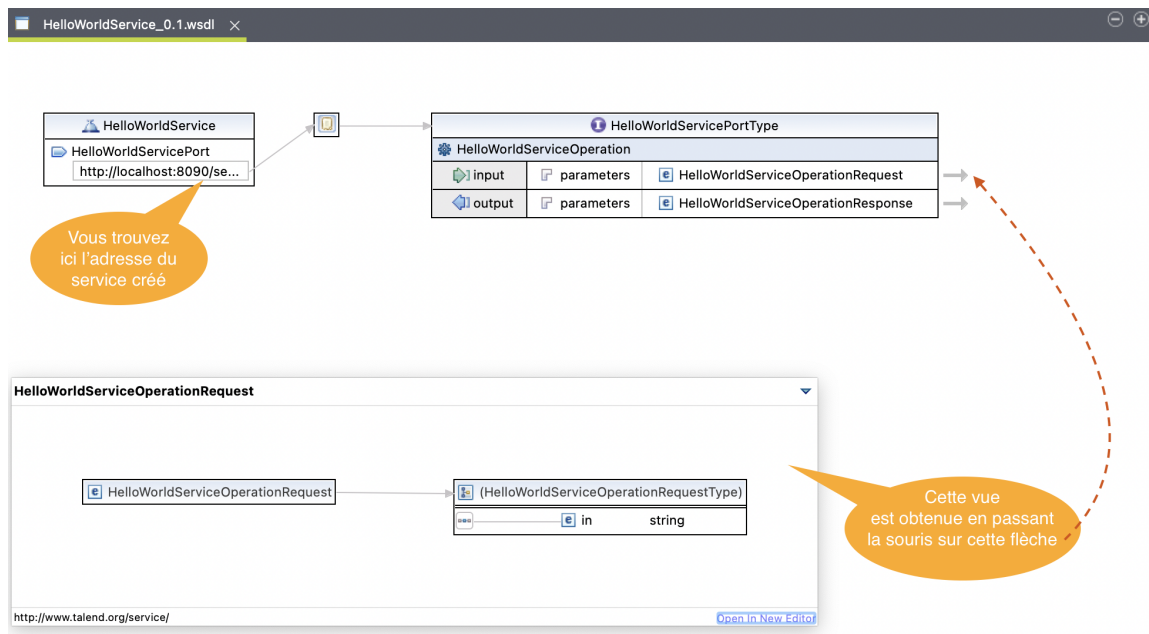
Nous ne pouvons pas configurer un ESB avant de savoir créer, déployer et exécuter des services web avec les outils Talend ESB. Nous allons donc commencer avec le traditionnel Hello World. Pour cela, il faut commencer par créer un projet de votre choix. Dans toute cette partie, nous nous trouverons dans la perspective *Integration*.

Créer le service SOAP

Pour créer un nouveau service de type SOAP:

- Clic-droit sur *Services* de votre Repository, et choisir *Create Service*. Appeler le service *HelloWorldService*. Cliquer sur *Suivant*.
- On vous propose soit de créer un nouveau WSDL, soit de choisir un WSDL existant. Dans notre cas, nous créons un nouveau WSDL. Cliquer sur *Terminer*.

- Un service simple qui reçoit une chaîne de caractères et en produit une autre est créé. Une vue graphique de son WSDL s'affiche.



Configurer le service SOAP

Pour pouvoir configurer votre service, il faut créer un Job. Mais d'abord:

- Importer le WSDL de votre service dans votre repository. Pour cela, clic droit sur *HelloWorldService*, et choisir *Importer les schémas WSDL*. Vous retrouverez votre WSDL dans la partie *Metadonnées -> Fichier XML*.
- Créer un nouveau Job pour votre service. Pour cela, clic-droit sur l'opération *HelloWorldServiceOperation* (sous Services) et choisir *Assign Job*.
- Modifier votre Job pour qu'il ait l'allure suivante (Le *tLogRow* nous permettra d'afficher le résultat du service exécuté sur la console avant de l'envoyer au consommateur):

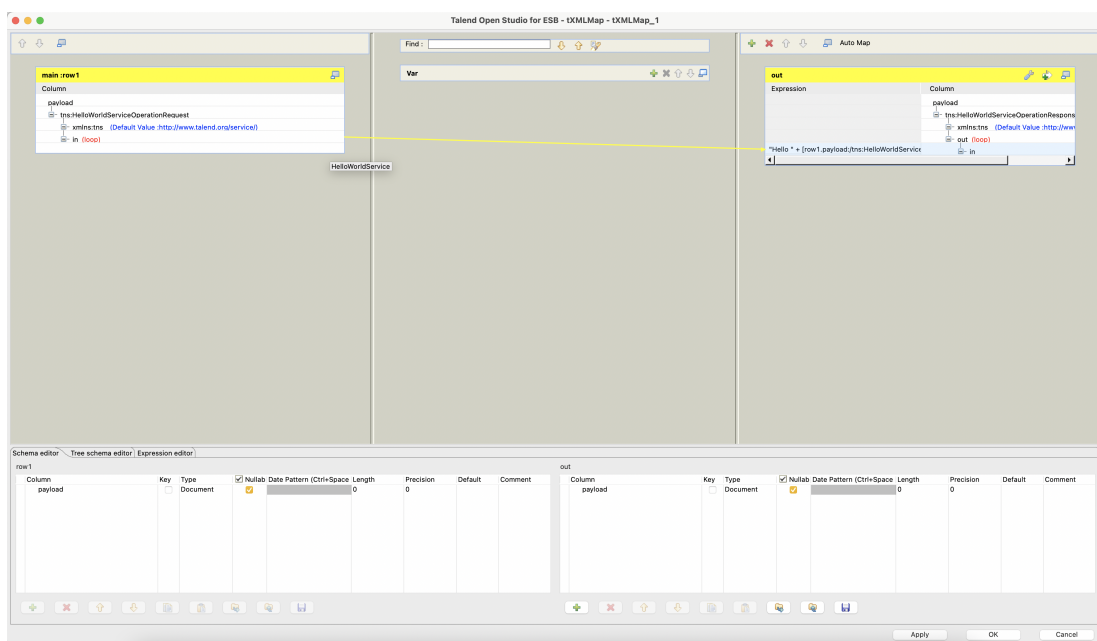


Astuce

Il est conseillé de relier d'abord les composants *tXMLMap* et *tESBProviderResponse_1*, puis d'insérer le *tLogRow*. On vous demandera en reliant les deux premiers composants : *Récupérer le schéma du composant cible?*. Cliquer sur *Oui*.

- Configurer votre *tXMLMap* pour que le *in* de la requête soit transmise au *out* de la réponse, en lui concaténant le célèbre "Hello". Pour cela:

- Double clic sur votre XML Map.
- Clic-droit sur *payload* de l'entrée, et cliquer sur *Import from Repository*.
- Choisir le *HelloWorldServiceOperationRequest* correspondant au fichier WSDL que vous avez généré.
- Refaire les mêmes étapes pour le payload de la sortie, en choisissant *HelloWorldServiceOperationResponse*.
- Relier le *in* de la requête avec le *out* de la réponse (créer l'entrée comme sous-élément de la réponse)
- Modifier l'expression du *out* en ajoutant la chaîne **"Hello "** avant la valeur *in* de l'entrée.
- Le résultat de la XMLMap devrait ressembler à ce qui suit:



- Sauvegarder et quitter.
- Lancer votre Job (cela permettra de publier votre service web sur le port 8090). Vérifier que votre fichier WSDL existe bien.

Tester le service SOAP

Il est possible de tester votre service de plusieurs manières. L'une d'elles est d'utiliser un outil léger de test appelé *SOAPUI*.

- Lancer SOAPUI
- Cliquer sur l'icône SOAP en haut de la fenêtre principale
- Donner un nom au projet (par exemple Helloworld) et entrer l'adresse du fichier WSDL du service, comme suit:

New SOAP Project
Creates a WSDL/SOAP based Project in this workspace

Project Name:

Initial WSDL:

Create Requests: ☒ Create sample requests for all operations?

Create TestSuite: ☐ Creates a TestSuite for the imported WSDL

Relative Paths: ☐ Stores all file paths in project relatively to project file (requires save)

- Double cliquer sur la requête générée, et remplir le ? par un nom de votre choix.
- Cliquer sur la flèche verte. Le résultat devra ressembler au suivant:

Request 1

http://localhost:8090/services/HelloWorldService.

Raw

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <ser:HelloWorldServiceOperationRequest>
      <in>Mehdi</in>
    </ser:HelloWorldServiceOperationRequest>
  </soap:Body>
</soap:Envelope>
```

Raw

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <tns:HelloWorldServiceOperationResponse xmlns:tns="http://www.talend.com">
      <out>
        <in>Hello Mehdi</in>
      </out>
    </tns:HelloWorldServiceOperationResponse>
  </soap:Body>
</soap:Envelope>
```

Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0) Headers (4) Attachments (0) SSL Info WSS (0) JMS (0)

response time: 12ms (266 bytes) 8:20

Consommateur du Webservice SOAP

Nous allons maintenant créer un consommateur pour notre service avec talend open studio.
Pour cela:

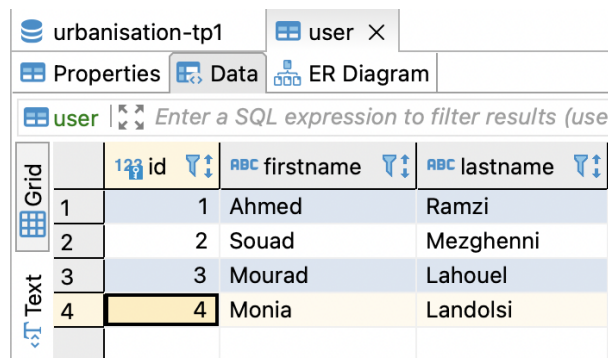
- Créer un nouveau Job, que vous appellerez *HelloWorldServiceConsumer*.
- Concevez votre job de manière à ce qu'il ait l'allure suivante:

Nous allons maintenant montrer comment exposer un service REST pour interroger une base de données.

Création de la base de données

Nous allons commencer par créer une base de données (MySQL dans mon cas), appelée *urbanisation-tp1* avec une table, que nous appellerons *user*. Cette table contient les champs *id*, *firstname* et *lastname*. Remplir ensuite la base à votre guise, de manière à avoir au moins 4 entrées.

Elle devra ressembler à ce qui suit:



urbanisation-tp1		user X	
Properties		Data	ER Diagram
user		Enter a SQL expression to filter results (use	
Grid	id	firstname	lastname
1	1	Ahmed	Ramzi
2	2	Souad	Mezghenni
3	3	Mourad	Lahouel
4	4	Monia	Landolsi

Ajout de la connexion à la base avec Talend

Pour configurer une connexion à cette base de données avec Talend, suivre les étapes suivantes:

- Dans les Métadonnées, sous *Connexions aux bases de données*, clic-droit, puis choisir: *Créer une connexion*.
- Configurer votre connexion. Voici un exemple:

Database Connection

New Database Connection on repository – Step 2/2

Define the connection parameters

DB Type: MySQL

Db Version: MySQL 8

String of Connection: jdbc:mysql://localhost:3306/urbanisation-tp1?noDatetimeStringSync=true&enabledTLSProtocols=TLSv1.2,TLSv1.1

Login: root

Password:

Server: localhost

Port: 3306

DataBase: urbanisation-tp1

Additional parameters: noDatetimeStringSync=true&enabledTLSProtocols=TLSv1.2,TLSv1.1,TLSv1

Test connection v

Export as context Revert Context

[How to install a driver](#)

< Back Next > Cancel Finish

- Une fois la connexion créée, importer son schéma. Pour cela, clic-droit sur Metadonnées -> Connexions... -> et choisir: *Récupérer le schéma*.
- Sélectionner la table *user*, et vérifier que les champs sont bien chargés dans la partie Schéma.

Création du service REST

Nous allons maintenant créer le service REST. Pour cela, créer un nouveau job, qu'on appellera *DBService*, puis glisser les composants suivants:

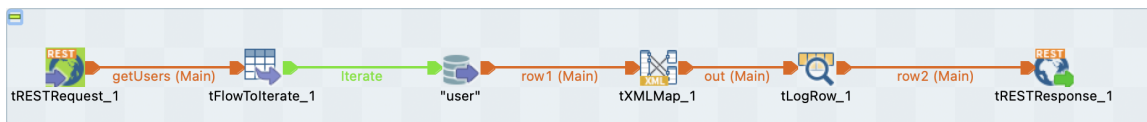
- **tRestRequest** : Pour définir la requête REST que le client doit appeler
- **user** : Table de la base de données. Dans la nouvelle connexion à la base de données que vous avez créé, sous *Schémas des tables*, glisser la table *user* vers le Job, puis choisir tMySQLInput dans la fenêtre de choix qui apparaît.

Remarque

Je choisis *tMySQLInput* car, dans mon cas, c'est une base de données MySQL, et je veux juste lire son contenu, je vais donc y accéder en entrée (d'où le *Input*).

- **tFlowTolterate** : Pour effectuer une itération sur les données d'entrée et générer des variables globales.
- **tXMLMap** : Permet de router et transformer les flux entrants de la base de données vers le résultat de la requête.
- **tRestResponse** : Pour définir la réponse à envoyer à l'utilisateur suite à sa requête.
- **tLogRow** : Pour le log, bien sûr.

Le job aura l'allure suivante:



Configuration du service REST

Nous désirons configurer le service de manière à ce que, quand un consommateur appelle l'URI: `http://localhost:8088/users?from=1&to=3`, le service retourne une réponse contenant les utilisateurs (id, nom et prénom) de la base de données dont les ids figurent entre 1 et 3.

CONFIGURATION DE TRESTREQUEST

tRestRequest devra être configuré comme suit:

- La valeur de *Endpoint URL* devra être: `"http://localhost:8088/users"`
- Si vous avez connecté le tRestRequest avec le tFlowTolterate avec un lien appelé *getUsers*, vous devriez le retrouver dans la case *REST API Mapping*. Sinon, créez-le.
- Garder les informations par défaut de ce mapping (méthode GET, URI /, Produit XML ou JSON).
- En cliquant sur *getUsers*, un bouton avec trois petits points apparaît. Cliquez dessus.
- Ajouter les deux colonnes *from* et *to* représentant les deux paramètres de la requête. Prenez soin à ce que:
 - Leur type soit *int*
 - Leurs valeurs par défaut soient respectivement 1 et 3.

Remarque

Ces valeurs seront utilisées dans le cas où le consommateur n'introduit pas de paramètres.

- Leur commentaire ait la valeur: *query*

Remarque

Cela indique que ces champs sont des paramètres de requête, pas définies dans le Path.

CONFIGURATION DE USER

Puisque le composant *user* a été créé à partir de la connexion à votre base MySQL, il contient déjà les informations de connexion nécessaires.

Il suffira dans notre cas de:

- Cliquer sur *Guess Schema* pour charger le schéma de la base.
- Changer la requête pour qu'elle soit comme suit:

```
"SELECT * FROM `user` where id>="+globalMap.get("getUsers.from")+  
    " and id<="+globalMap.get("getUsers.to")
```


Remarque

globalMap est une variable globale permettant de stocker les informations de la requête, comme par exemple ses paramètres. Ici on suppose que le nom de votre lien est *getUsers*. Si ce n'est pas le cas changez-le dans la requête.

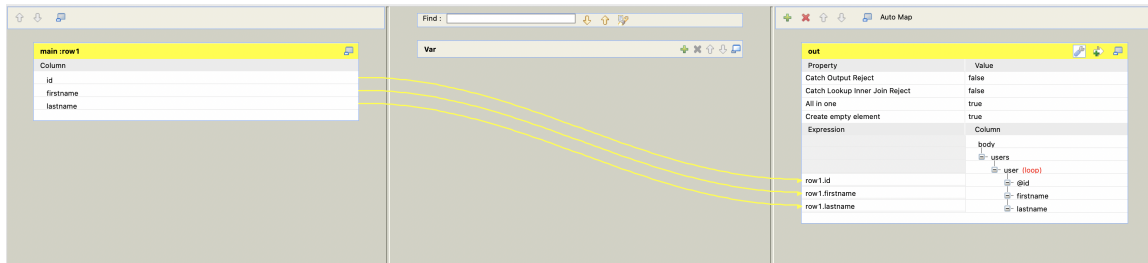
CONFIGURATION DE TXMLMAP

Cliquer deux fois sur la *tXMLMap* pour la configurer.

- Dans la colonne de droite, ajouter (si ce n'est déjà fait) une colonne intitulée *body* dont le type est *Document*.
- Cette colonne contient un élément *root*. Renommer cet élément pour *users*.
- Ajouter un sous-élément à *users* appelé *user*.
- Définir cet élément comme *loop Element*.
- Glisser-déplacer l'id de la colonne en entrée vers le *user*. Créez-le comme attribut du noeud cible.
- De même pour le *firstname* et *lastname*, qui seront, eux, des sous-éléments du noeud *user*.

- Dans la colonne de droite, cliquer sur la petite clef à molette (). Mettre la valeur de "All in one" à *true*. Cela permettra à toutes les données XML d'être écrites dans un seul flux.

La configuration finale sera donc comme suit:



Indication

La configuration précédente va générer une réponse de la forme suivante:

```
<users>
  <user id=1>
    <firstname> flen </firstname>
    <lastname> fouleni </lastname>
  </user>
  <user id=2>
    <firstname> flena </firstname>
    <lastname> foulenia </lastname>
  </user>
</users>
```

Les autres composants devront rester tels qu'ils sont par défaut. Il suffira maintenant de lancer le service, en cliquant sur *Exécuter*.

Tester le Service

DANS UN NAVIGATEUR

Pour tester le service, il suffit d'ouvrir un navigateur, et de taper la requête de votre choix.

Par exemple, la requête suivante : `http://localhost:8088/users?from=2&to=4` donnera:

```
<users>
  <user id="2">
    <firstname>Souad</firstname>
    <lastname>Mezghenni</lastname>
  </user>
  <user id="3">
    <firstname>Mourad</firstname>
    <lastname>Lahwel</lastname>
  </user>
  <user id="4">
    <firstname>Monia</firstname>
```

```
<lastname>LandoIsi</lastname>
</user>
</users>
```

Si aucun paramètre n'est indiqué: `http://localhost:8088/users` cela donnera:

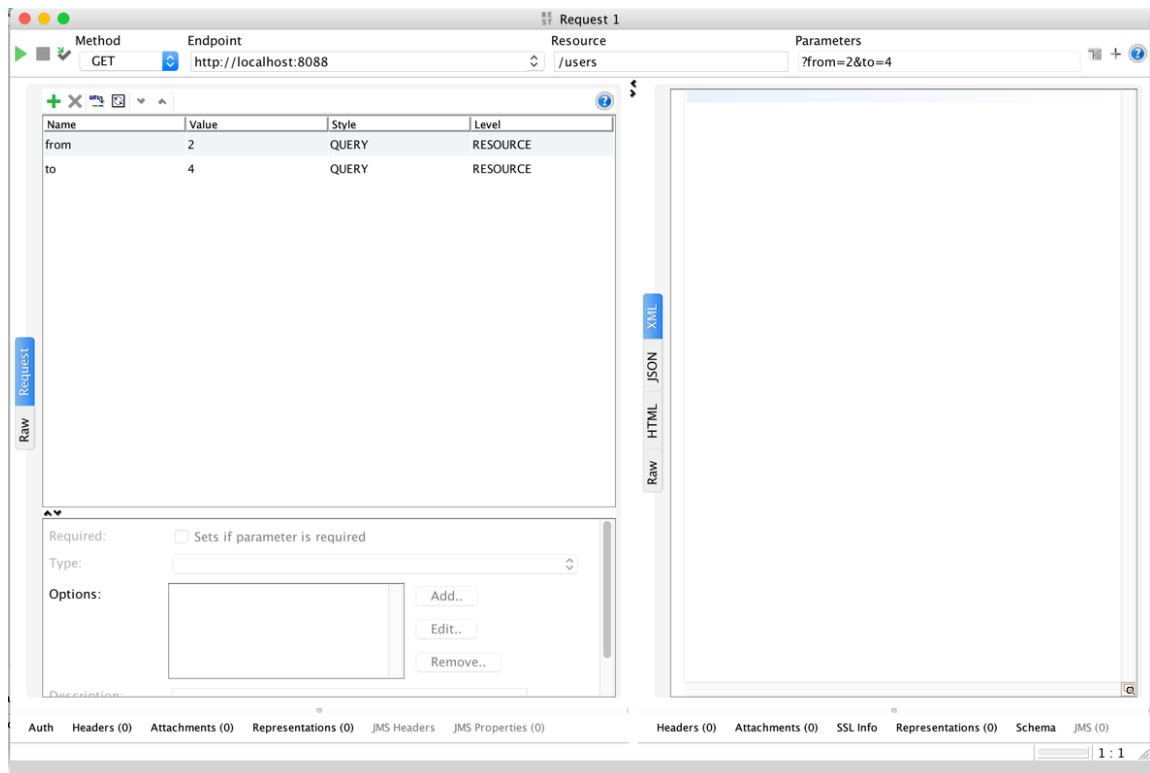
```
<users>
  <user id="1">
    <firstname>Ahmed</firstname>
    <lastname>Ramzi</lastname>
  </user>
  <user id="2">
    <firstname>Souad</firstname>
    <lastname>Mezghenni</lastname>
  </user>
  <user id="3">
    <firstname>Mourad</firstname>
    <lastname>Lahwel</lastname>
  </user>
</users>
```

AVEC SOAPUI

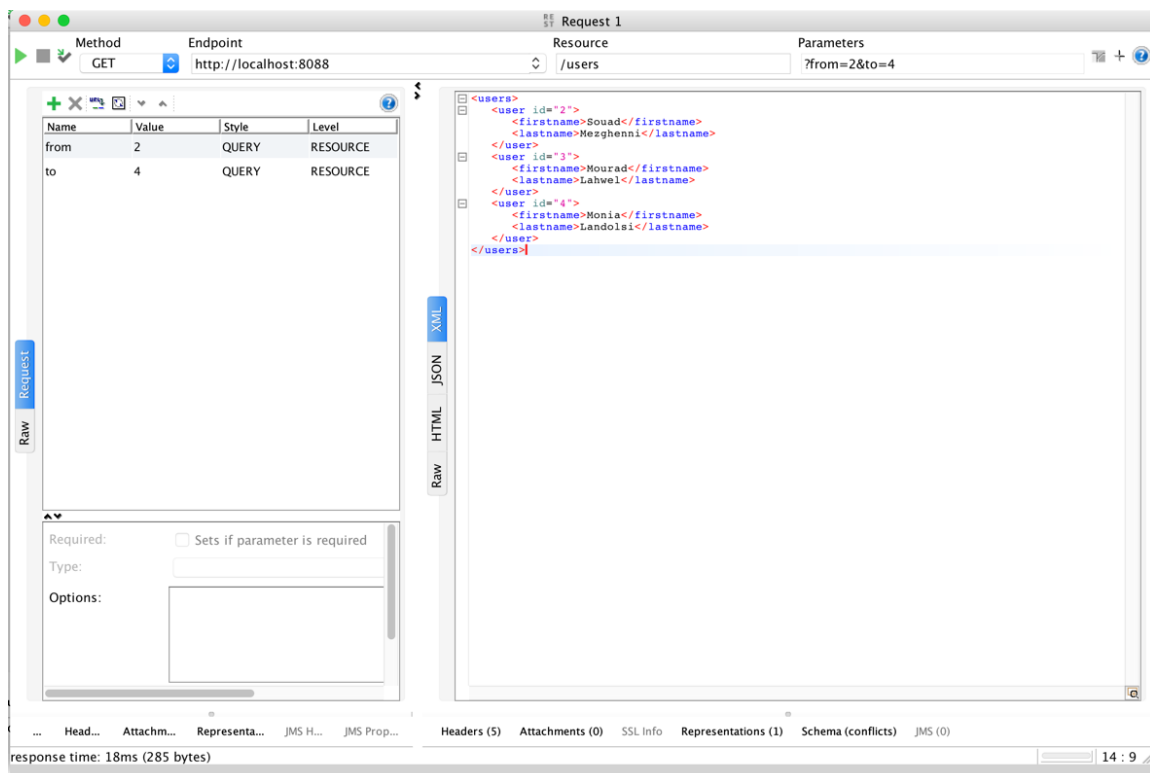
Tester le service REST

Il est possible de tester votre service REST avec *SOAPUI*.

- Lancer SOAPUI
- Cliquer sur l'icône REST en haut de la fenêtre principale
- Entrer l'URI que vous désirez tester: `http://localhost:8088/users?from=2&to=4`
- La fenêtre suivante devrait apparaître:

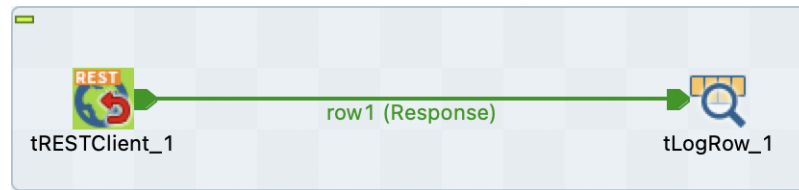


- Cliquer sur la flèche verte. Le résultat devra ressembler au suivant:



Consommateur du Webservice REST

Pour créer un consommateur pour le web service REST avec Talend, il suffit de créer le Job suivant:



Configurer le *tRestClient* comme suit:

Job(RESTConsumer 0.1)

Contexts(RESTConsumer)

Component

Run (Job RESTConsumer)

tRESTClient_1

Basic settings

Advanced settings

Dynamic settings

View

Documentation

URL

Relative Path

HTTP Method

Accept Type

XML

Query parameters

name	value
"from"	"2"
"to"	"4"

+

×

↑

↓

📄

📁

Input Schema

Built-In

Edit schema

...

Response Schema

Built-In

Edit schema

...

Error Schema

Built-In

Edit schema

...

Exécuter. Le résultat devrait ressembler à ceci:

[illegible]

Last update: 2022-08-19