



# Back End Basic training

---

Low Code Studio



## How to use this tutorial ?

Welcome to your first journey with Convertigo Low Code Studio. Let's explore its many features.



## Concepts & Definitions

Convertigo uses many concepts you may not be familiar with. Find answers with this icon.



## Practice time

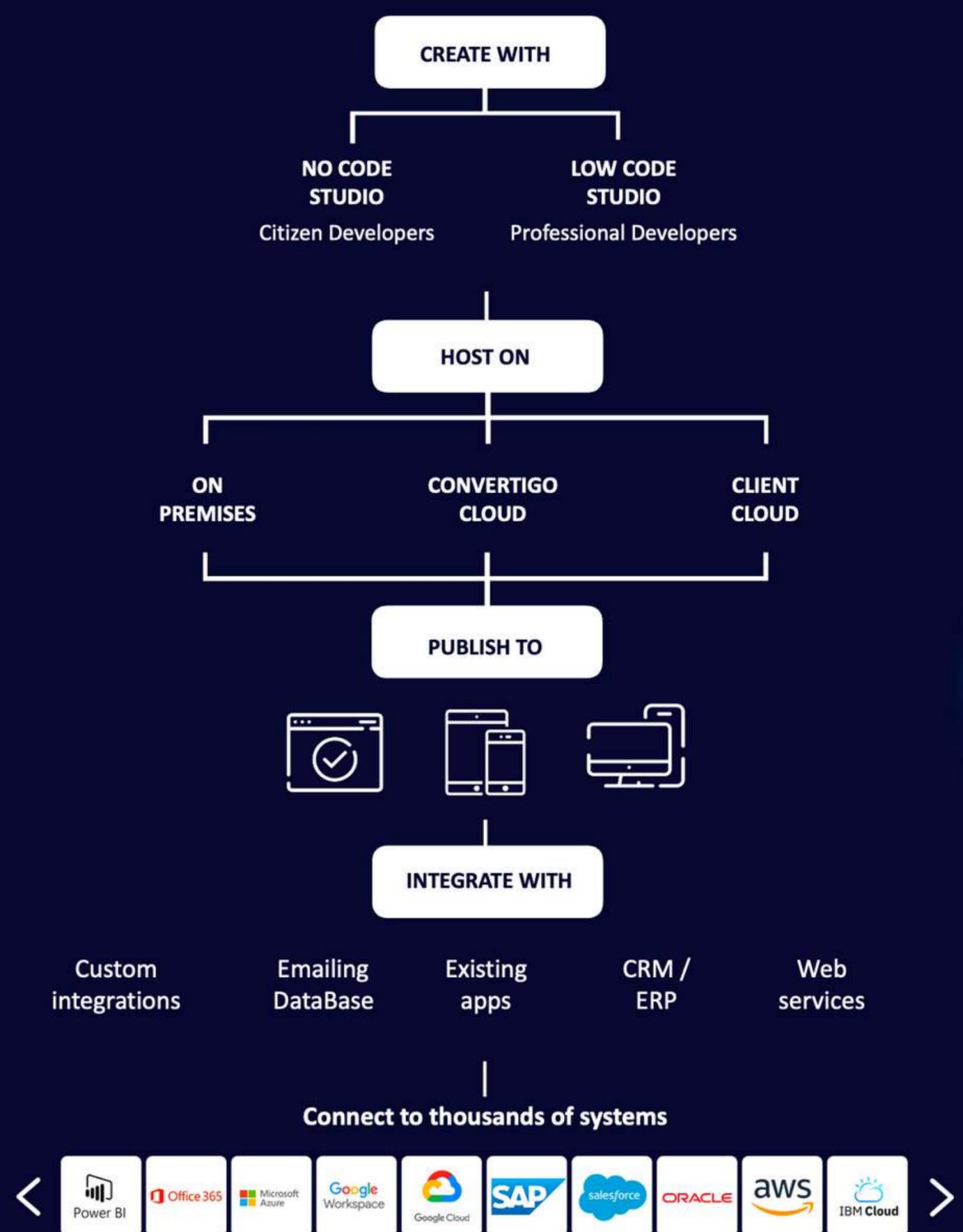
You prefer to skip the concepts and start by practice. Go straight to this icon.





# What is Convertigo Low code Platform ?

- > Full Stack
- > Low Code
- > Open Source
- > Application Development Platform



What  
can you do  
with

Convertigo  
Low code  
Studio ?



- ✓ Connect to back end systems with **Connectors**
- ✓ Exchange data with the backend using **Transactions**
- ✓ Define backend flows and business logic with **Sequences**
- ✓ Create web and mobile user interfaces with **Pages** and **UI Components**
- ✓ Create **iOS, Android, Progressive Web Apps** and **Web applications** from the same project
- ✓ Define and execute **Test cases**
- ✓ Share your projects with **Git versioning**

# Table of Contents

## 1 – INTRODUCTION

Overview of the studio.

## 2 – GETTING STARTED

How to install and configure the studio.

## 3 – WEB SERVICES CONNECTORS & TRANSACTIONS

How to consume a Rest API.

## 4 – SEQUENCES

How to create a flow of actions.

## 5 – JAVASCRIPT SCOPE

How to handle JavaScript in the studio.

## 6 – ERROR MANAGEMENT

How to handle errors in the studio.

## 7 – COLLABORATION WITH GIT

How to share your projects with Git Versioning.

## 8 – TEST PLATFORM

How to test the backend.

## 9 – URL MAPPER

How to expose an API REST.

## 10 – NOCODE DATABASE

How to use the NoCode Database.

## 11 – LOGS

How to manage logs in the studio.

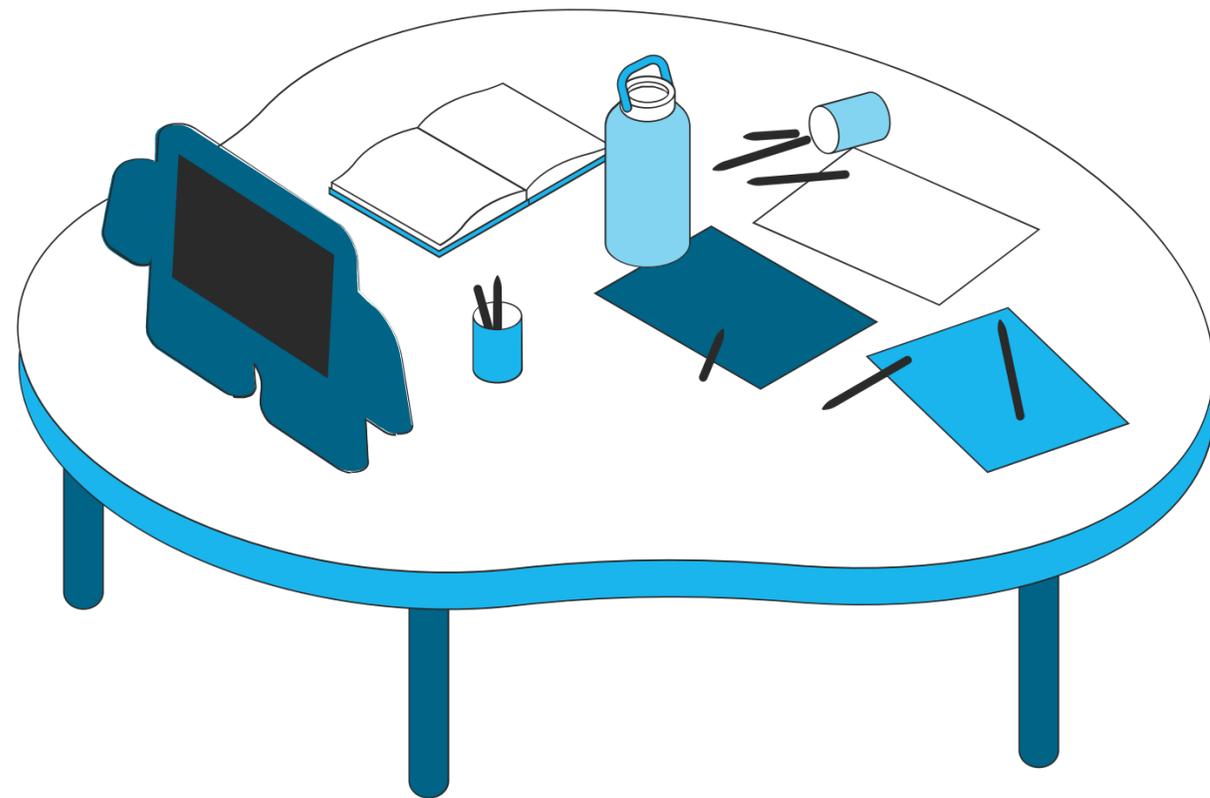
## 12 – AUTHENTICATION

How to manage authentication in the studio.

## APPENDIX

# 1 – Introduction

Overview of the studio.



**1.1** Technical knowledge

---

**1.2** Global architecture

---

**1.3** Convertigo Server

---

**1.4** Objects in Convertigo

---

**1.5** Back-end Objects

---

**1.6** Studio Interface

---

**1.7** Panels & Views

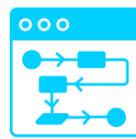
# 1.1 Technical knowledge

The following concepts are necessary for mastering the studio.



## WEB TECHNOLOGIES

- XML and XPath
- JavaScript & JSON
- HTTP requests
- REST API



## ALGORITHMS

- Pseudocode basics
- Loops, Conditional statements...



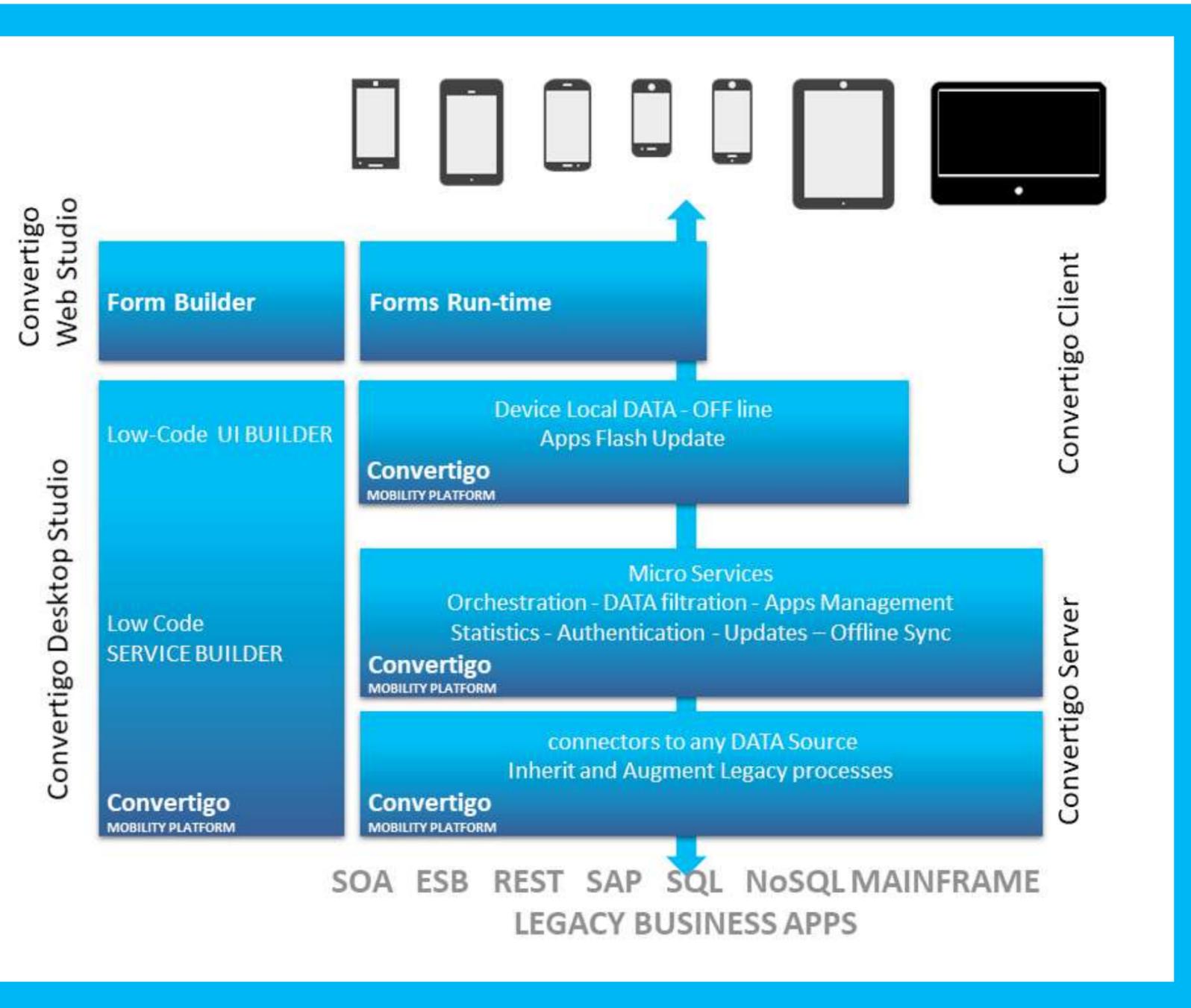
## DATABASES

- SQL basics
- NoSQL basics



# 1.2 Global architecture

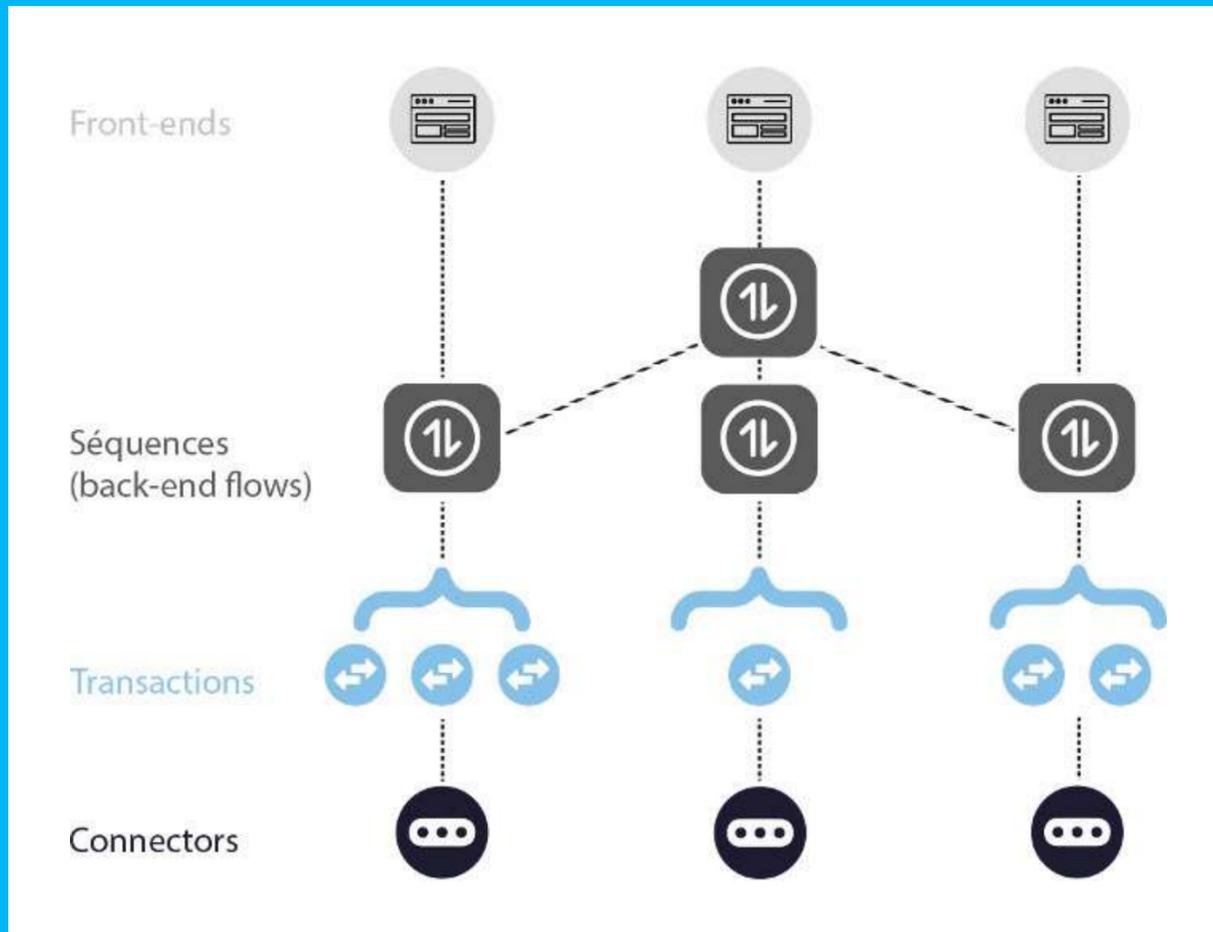
- Full Stack Low Code Platform
- Built on Docker Container Micro service Architecture
- Based on Open Standard technologies
  - (Eclipse, Node, Cordova, Angular, Java, Docker, Kubernetes, CouchDB, ...)
- Out of the box Connectors to Enterprise Data
- 100% Offline Data sync technology
- Interface with AI & ChatBots
- Includes 100% Web Studio for Form Builder



# 1.3 Convertigo Server



- ✓ Runs the **back-end** of the application
- ✓ Can be used to **deploy** as many apps as wanted
- ✓ Handles data in a **NoSQL database**
- ✓ Provides **connectors** to many **data providers** (SQL, Web services, Legacy apps running on mainframes...)
- ✓ Runs in **Docker container platforms** as a **Docker Image** (Cloud providers, Kubernetes on premises...)



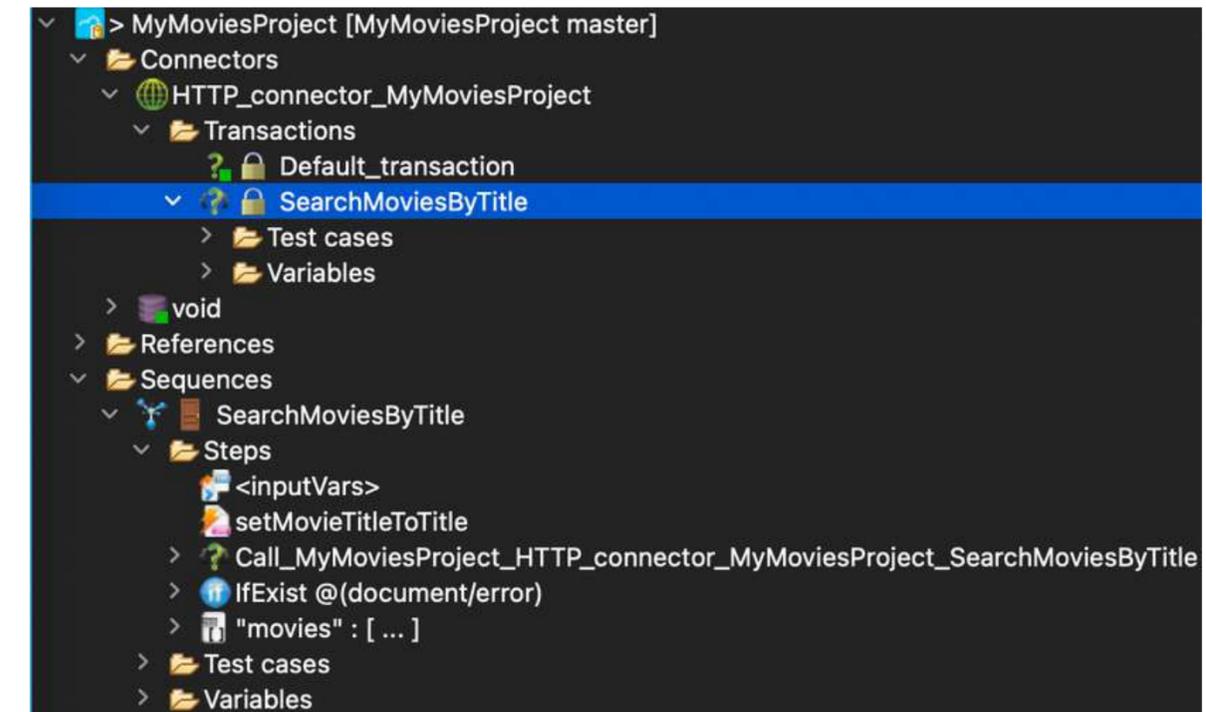
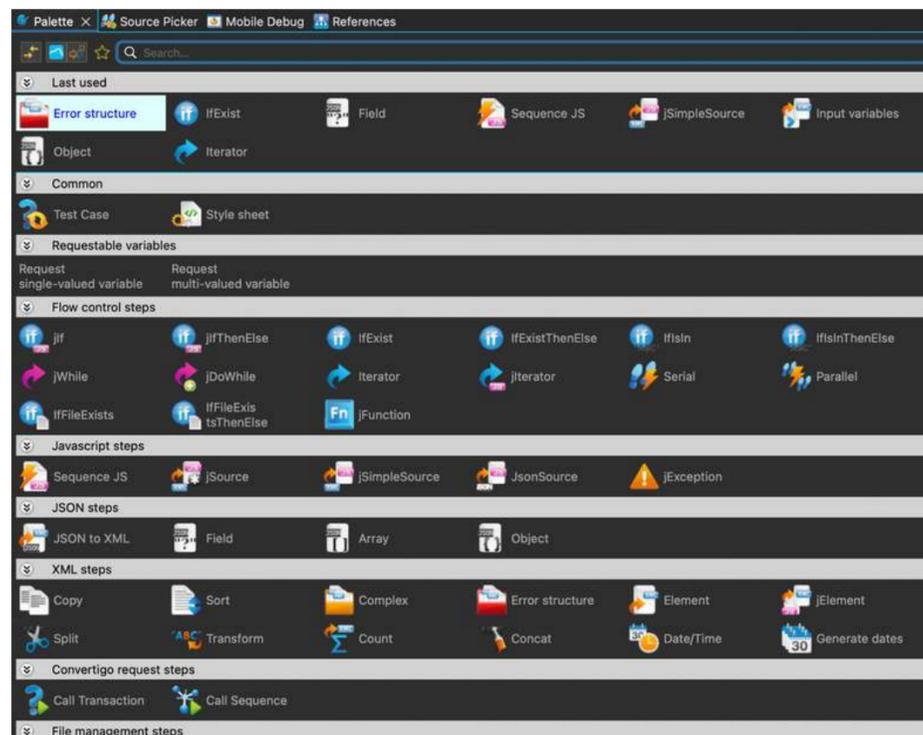
# 1.4 Objects in Convertigo

In Convertigo, **Objects** refer to **structured components** that **encapsulate data, functions, and properties**.

Objects are used to **represent and manipulate various elements** in Convertigo projects.

The objects are available in the **Palette view**.

A Convertigo project is **organized in a treeview**, where you **drop objects** dragged from the **palette**.



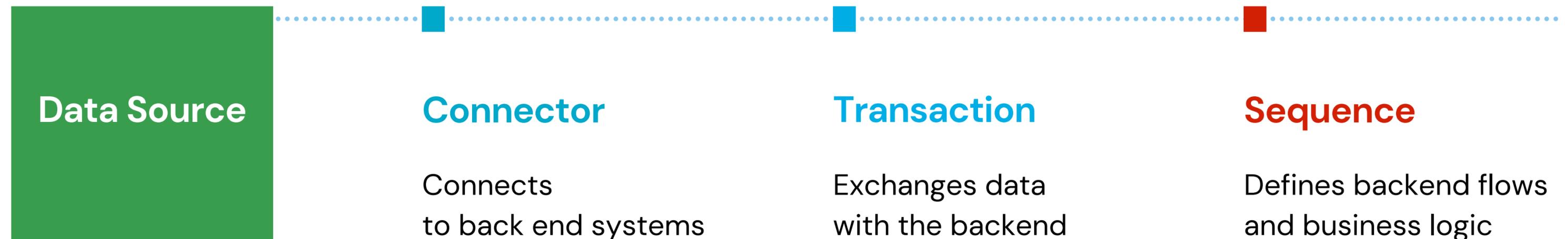
# 1.5 Back-end Objects

In a Convertigo project, **back-end objects** handle the **back end processing**.

There are **3 main back-end objects** : **Connector**, **Transaction**, and **Sequence**.

**Sequences interact with Connectors and Transactions**

to read and write data to Databases, WebServices or Third party applications.



# 1.6 Studio Interface

The studio interface is divided in **5 main panels**. Each one contains **several views**.

## PROJECTS PANEL

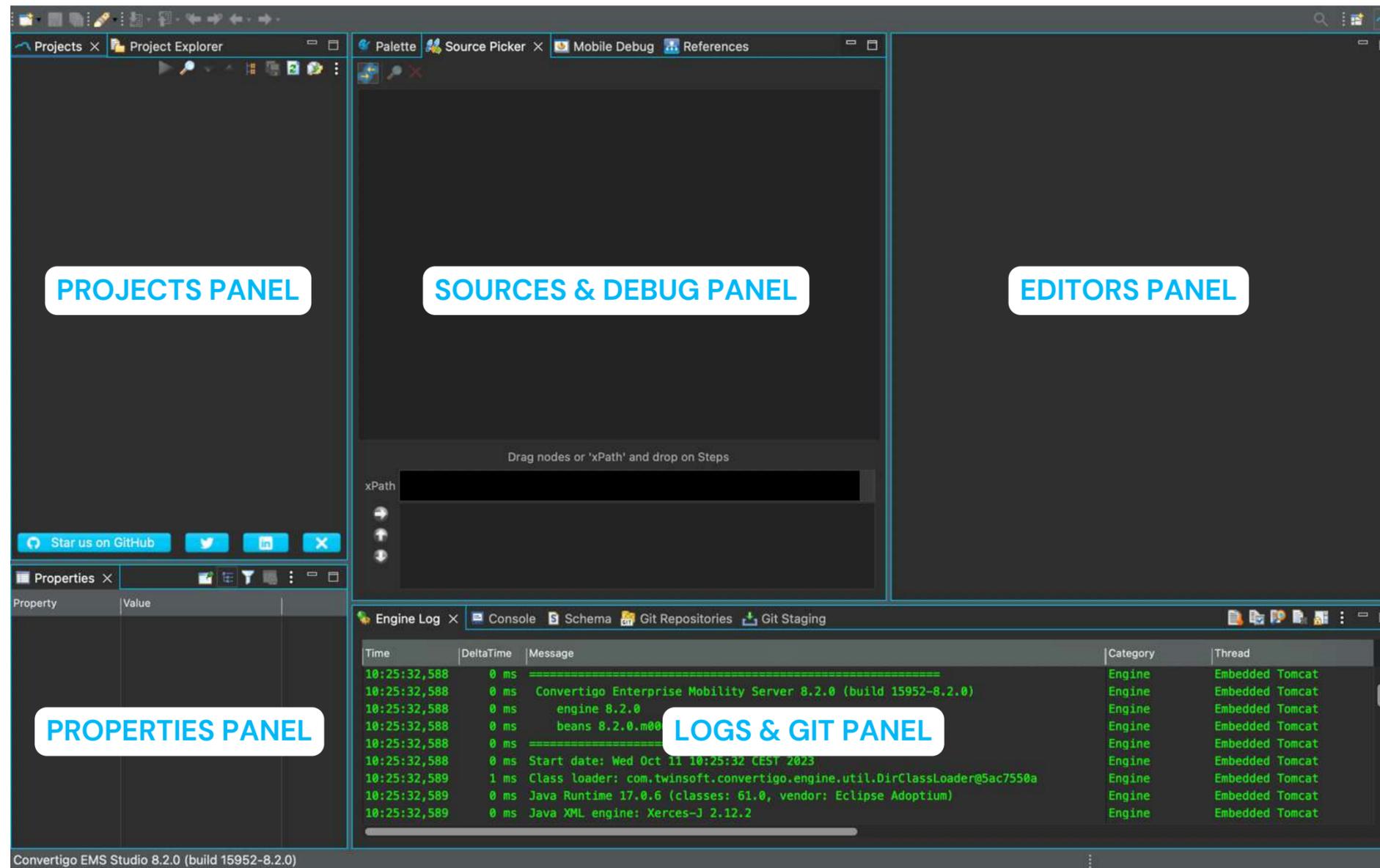
-  PROJECTS
-  PROJECT EXPLORER

## SOURCES & DEBUG PANEL

-  PALETTE
-  SOURCE PICKER
-  REFERENCES
-  MOBILE DEBUG

## PROPERTIES PANEL

-  PROPERTIES



## EDITORS PANEL

-  VISUAL APP VIEWER
-  CODE EDITORS
-  CONNECTORS & SEQUENCES RESPONSES

## LOGS & GIT PANEL

-  ENGINE LOG
-  CONSOLE
-  SCHEMA
-  GIT STAGING
-  GIT REPOSITORIES



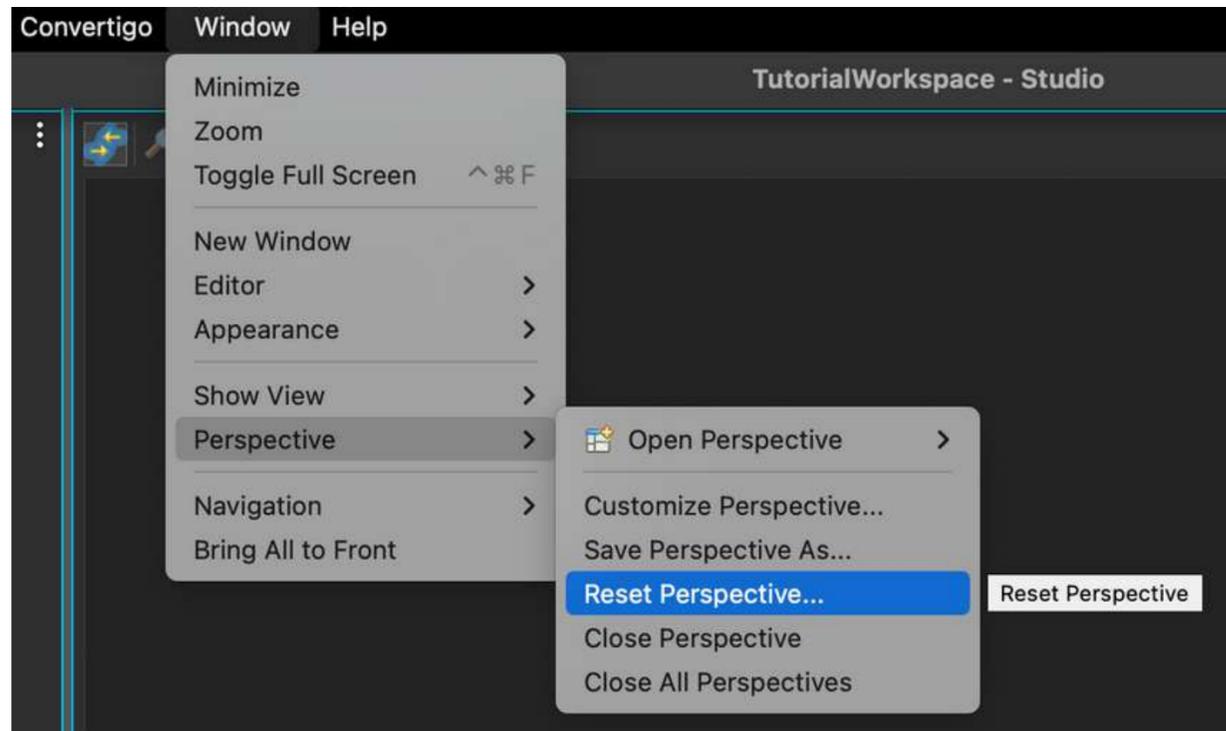
# 1.6 Studio Interface

The way views are organized is called a **perspective**.

Each view can be moved in other panels.

You can return to the original presentation or perspective

by clicking on **Window**, then selecting **Perspective**>, then selecting **Reset Perspective**.



# 1.7 Panels & Views

## Projects Panel



### PROJECTS

Displays the **projects in current workspace** and the **objects that compose them**.



### PROJECT EXPLORER

Displays the projects as **files representing project assets, projects definitions as yaml** (for advanced users only)

## Properties Panel



### PROPERTIES

Displays the **properties of the object** selected in Projects view.



# 1.7 Panels & Views

## Sources & Debug Panel



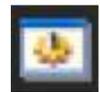
### PALETTE

Displays all Convertigo **backend and frontend objects**.



### SOURCE PICKER

Displays the **data sources for data binding** of the selected sequence step.



### MOBILE DEBUG

Displays the **debugger for the front end** part.



### REFERENCES

Displays **inside and outside project references** of the object selected in the Projects view.



# 1.7 Panels & Views

## Logs & Git Panel



### ENGINE LOG

Displays **Convertigo engine execution traces**.



### SCHEMA

Displays the **XSD schema** used and/or generated by the project (input and output).



### CONSOLE

Displays the **engine execution traces as text**.



# 1.7 Panels & Views

## Logs & Git Panel



### GIT REPOSITORIES

Displays the **Git Repositories** of the projects in your workspace.



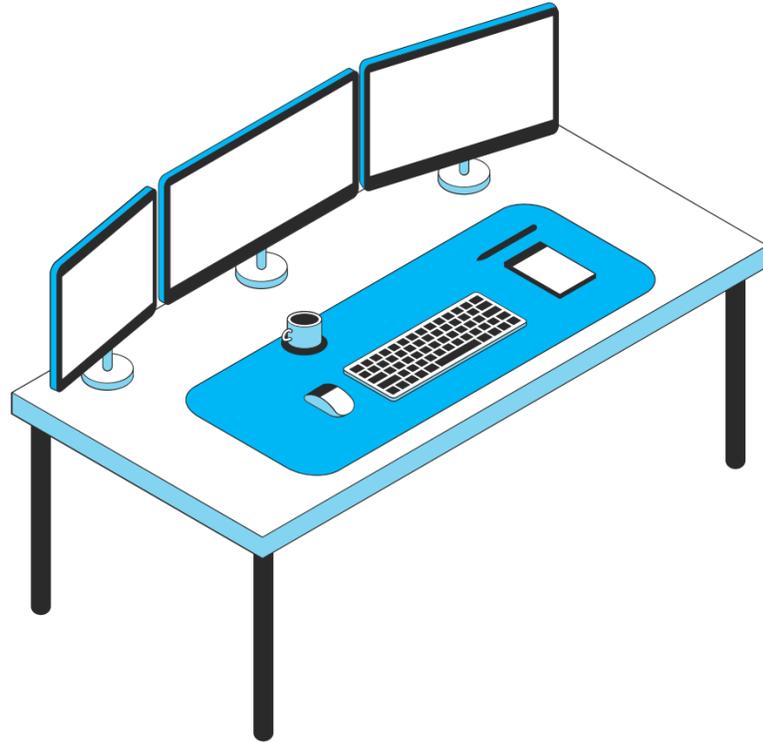
### GIT STAGING

Displays the files modified since the last commit and **Git management features**.



# 2 – Getting started

How to install and configure the studio.



**2.1** Minimum System Requirements

---

**2.2** Installation

---

**2.3** Workspace & Convertigo archives file

---

**2.4** Configuration

---

**2.5** Create a project

---

**2.6** Export a project

---

**2.7** Import a project

# 2.1 Minimum system requirements

The following minimum system requirements are necessary for installing the studio.



## WINDOWS

- Windows 10
- Windows 11



## LINUX

- Ubuntu
  - version 20.04 (LTS)
  - version 22.04 (LTS)
- Debian version 11.0



## MAC OS

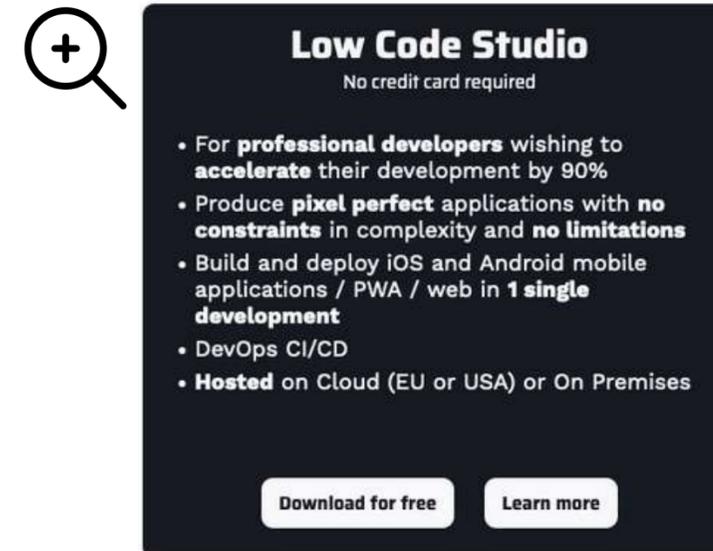
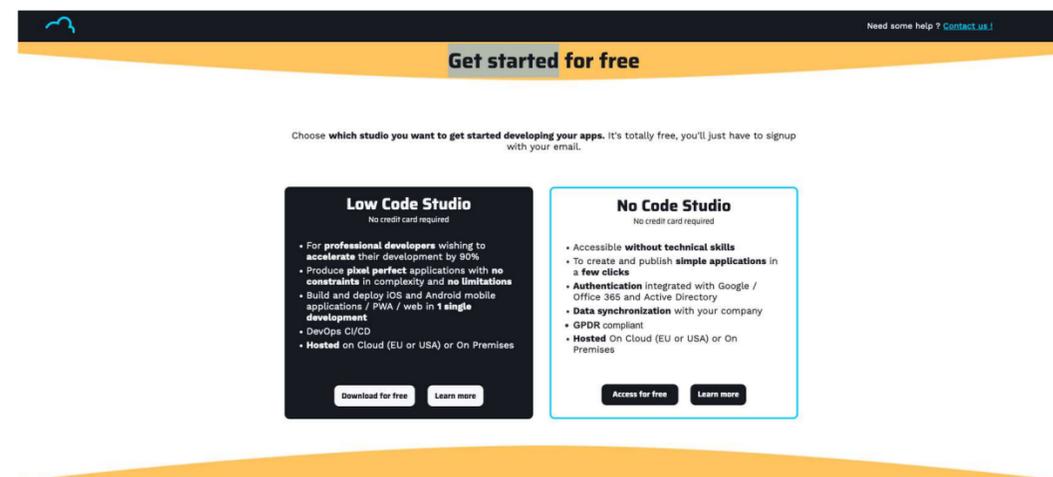
- Mac OS X
  - 10.5 (Leopard)
  - or greater
- Mac ARM



## 2.2 Installation

Go to the **Get started page** on <https://www.convertigo.com/get-started-page>.

Download the Low Code Studio package file for your operating system (Windows, Linux or Mac OS).



Open the package file and install the studio in a destination directory where you have the required permissions.

The installation package contains

- the Eclipse-based Convertigo Studio
- the embedded Convertigo Server with an Apache Tomcat application server



## 2.3 Workspace & Convertigo archives file

On first launch, you need to create a workspace for your projects.

A workspace is a directory where are saved

- Studio **configurations**
- Convertigo **projects**
- **Execution logs**

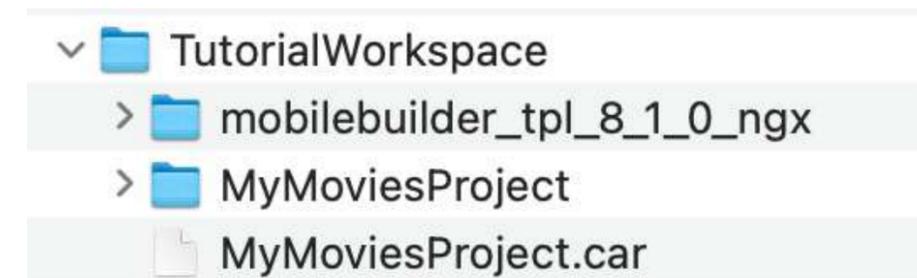
The workspace is **located outside** of the installation directory to **save your data**

if you need to **uninstall** or **re-install** the studio.

In Convertigo, the **import/export format** is **.car (Convertigo archives)** or **.zip**.

The **.car** is a zip file that contains all your project.

Example : A workspace with a Convertigo project and a .car file



## 2.4 Configuration

After installation, the Studio needs to be configured on first launch.

### CONFIGURATION PROCESS

- Step 1** Select a directory as workspace
- Step 2** Accept License
- Step 3** Complete the workspace creation
- Step 4** Configure proxy settings (Optional)
- Step 5** Register with Convertigo Cloud Trial
- Step 6** Welcome to Convertigo Low Code Studio



## 2.4 Configuration

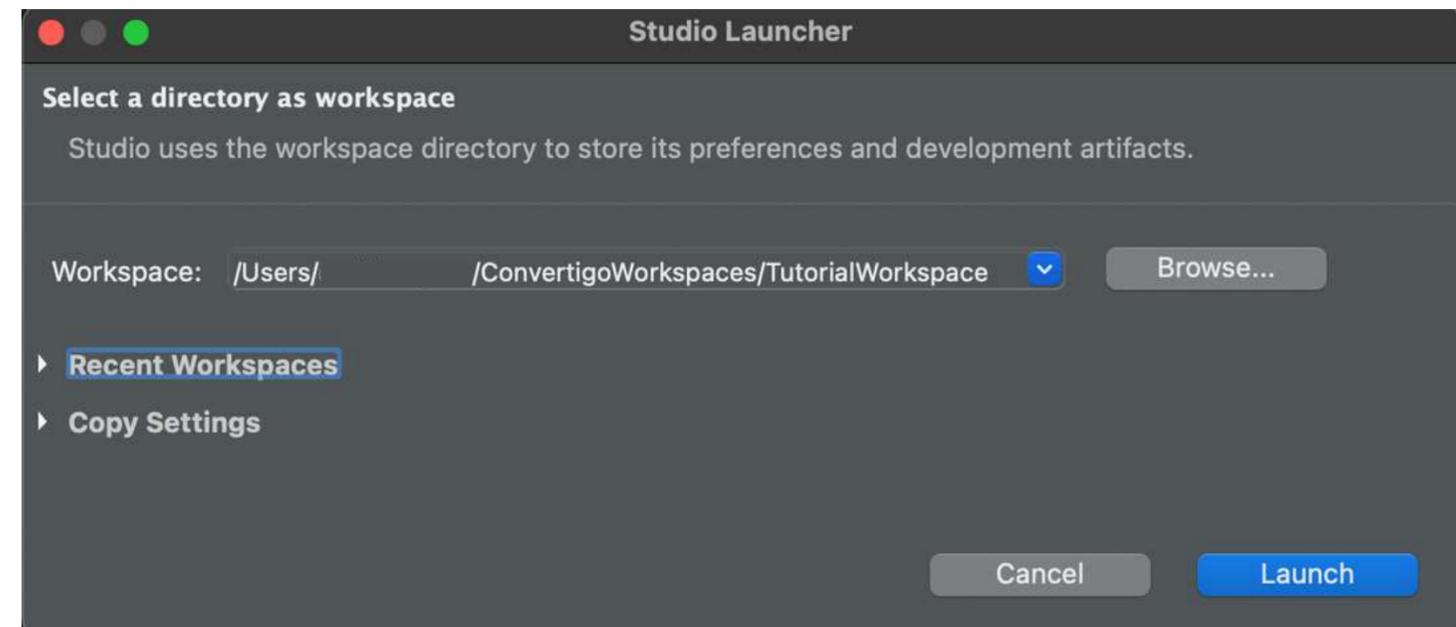
### Step 1 – Select a directory as workspace

This is the first time we are going to launch the studio. Let's start by creating a **workspace** for your projects.

Launch the studio and **select a folder** where your **workspace will be created**.

You can

- select an **existing** folder or **create** one.
- create **as many workspaces** as you want
- **wherever you want** on your computer.



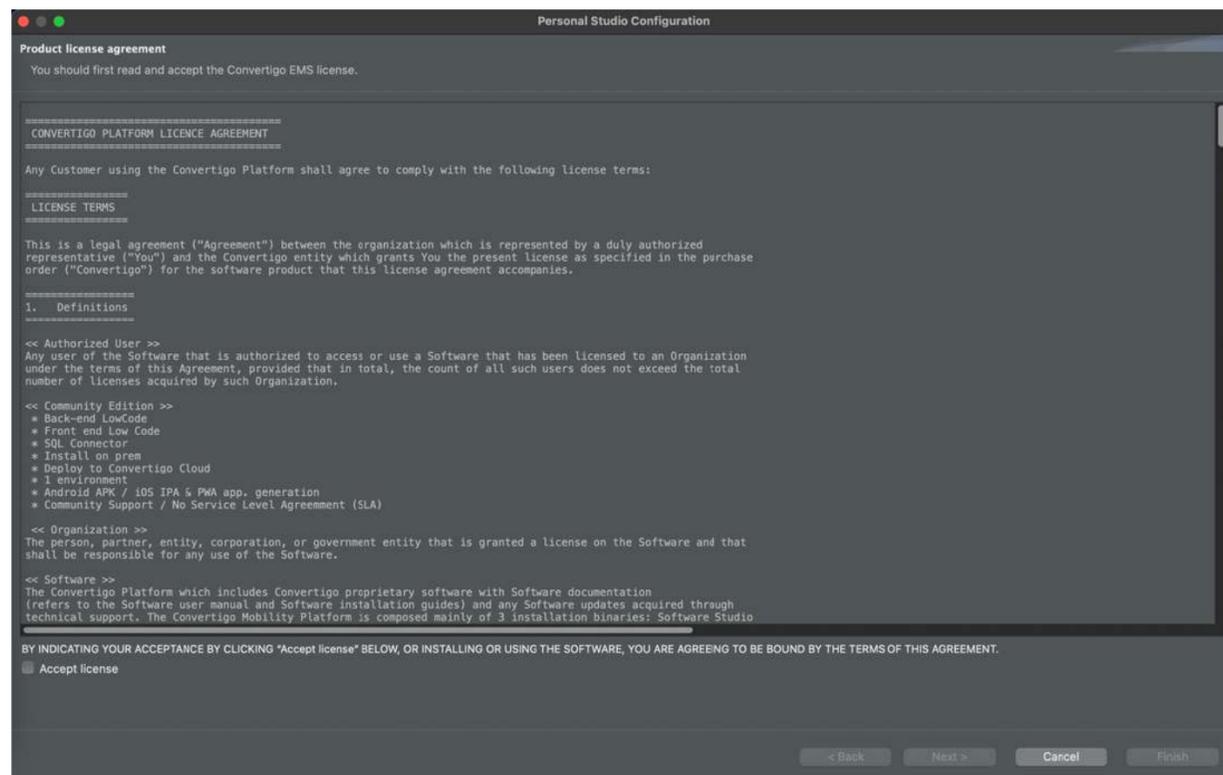
**Good practice:** create your workspace in your user folder on the same level as the Desktop and the Download folder – BUT NOT INSIDE THEM.



# 2.4 Configuration

## Step 2 - Accept License

In the window **Personal studio Configuration**,  
**Accept License** and click on **Next >**.



### Personal Studio Configuration

#### Product license agreement

You should first read and accept the Convertigo EMS license.

#### CONVERTIGO PLATFORM LICENCE AGREEMENT

BY INDICATING YOUR ACCEPTANCE BY CLICKING "Accept license"

Accept license

< Back

Next >

Cancel

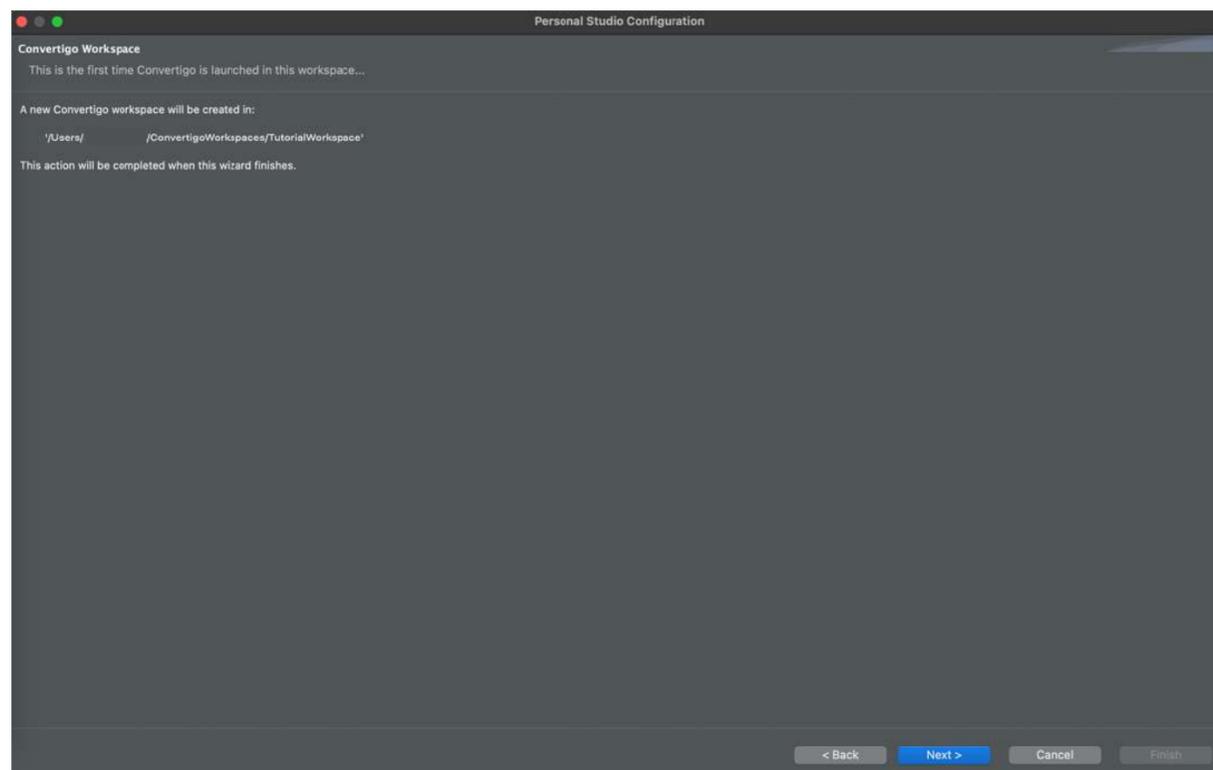
Finish



## 2.4 Configuration

### Step 3 - Complete the workspace creation

To complete the creation of your workspace, click on **Next >**



### Personal Studio Configuration

#### Convertigo Workspace

This is the first time Convertigo is launched in this workspace...

A new Convertigo workspace will be created in:

'/Users/ /ConvertigoWorkspaces/TutorialWorkspace'

This action will be completed when this wizard finishes.

< Back

Next >

Cancel

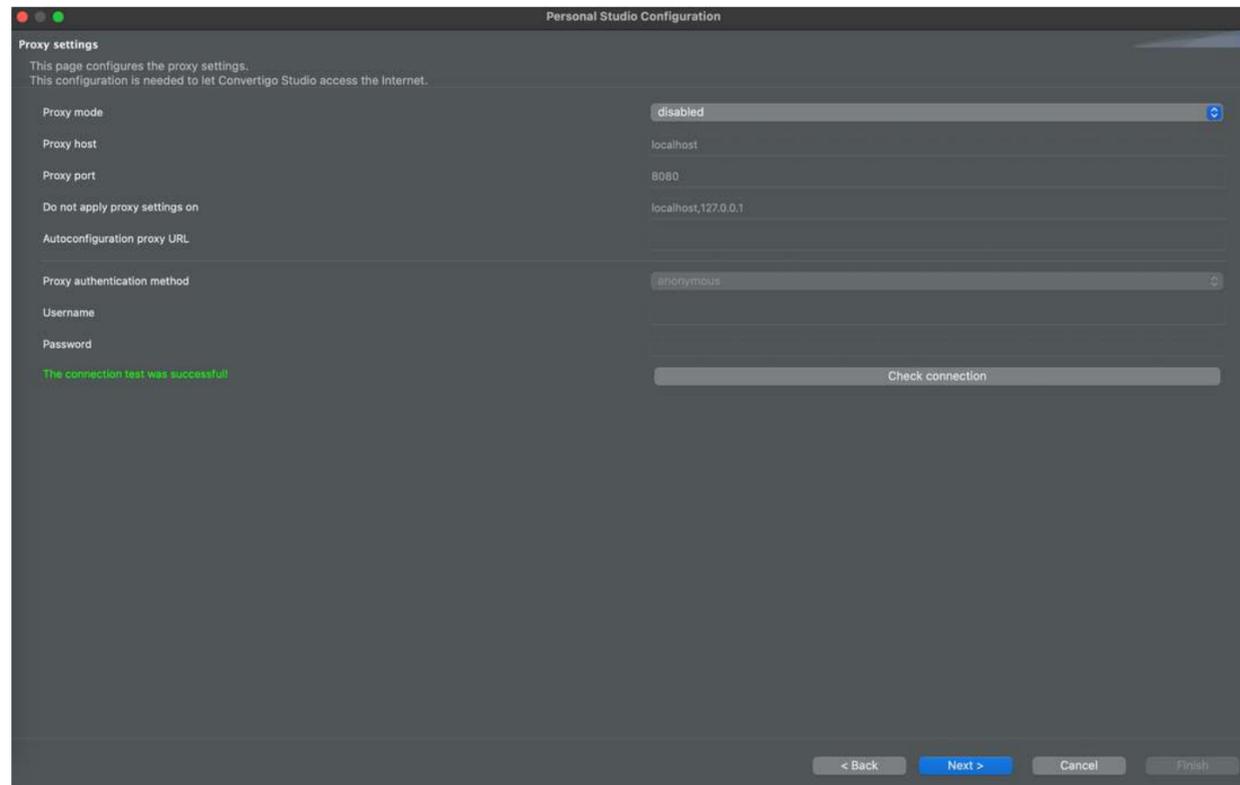
Finish



# 2.4 Configuration

## Step 4 – Configure proxy settings

Optional : You can **configure proxy settings** for Convertigo Studio to access the Internet

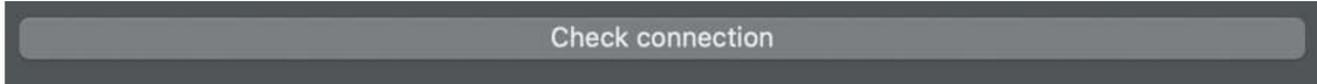


### Personal Studio Configuration

**Proxy settings**  
This page configures the proxy settings.  
This configuration is needed to let Convertigo Studio access the Internet.

Proxy mode	disabled
Proxy host	localhost
Proxy port	8080
Do not apply proxy settings on	localhost,127.0.0.1
Autoconfiguration proxy URL	
Proxy authentication method	anonymous
Username	
Password	

To check the connection, click on **Check connection**



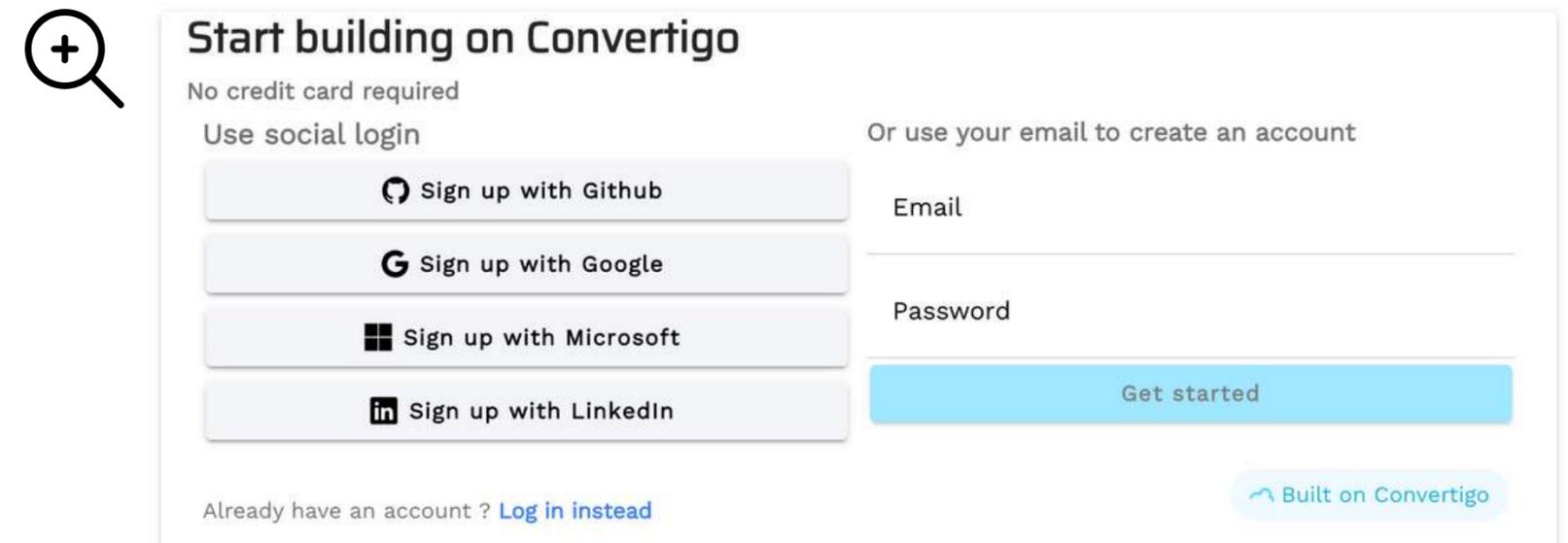
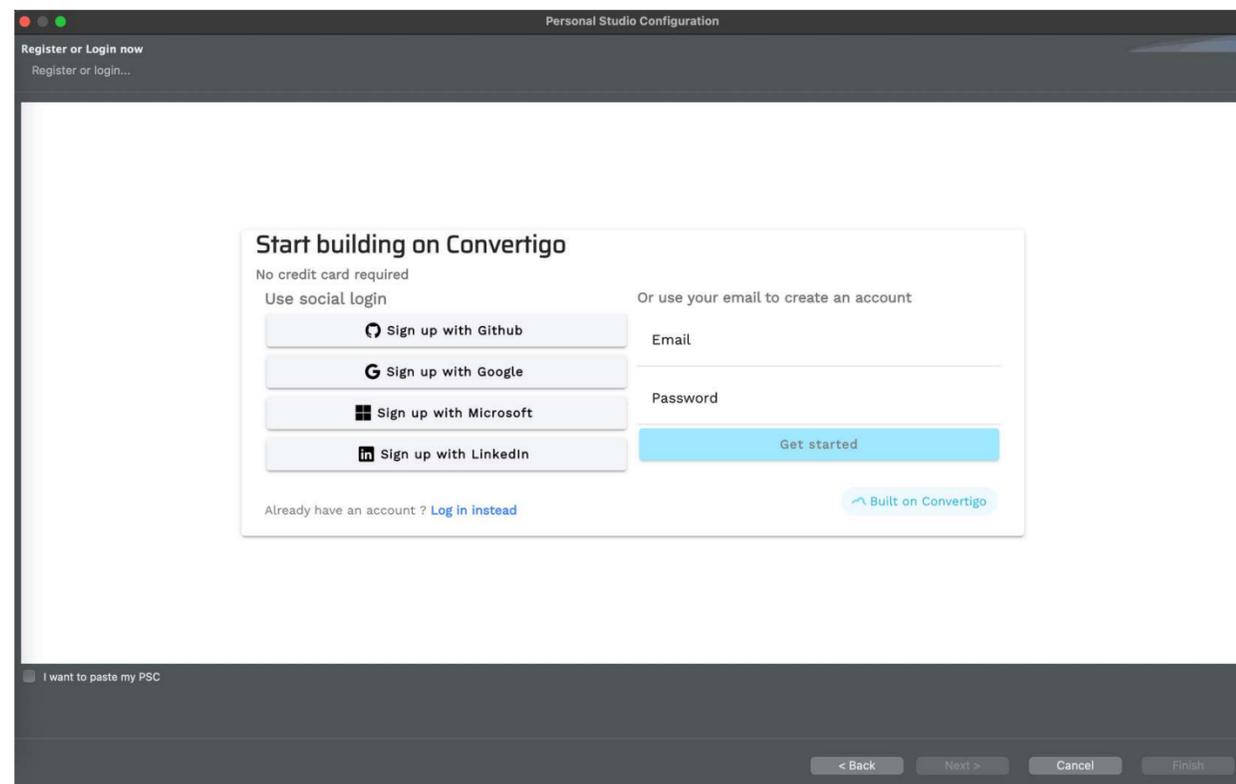
**The connection test was successful!**



# 2.4 Configuration

## Step 5 – Register with Convertigo Cloud Trial

Complete your registration with **Convertigo Cloud Trial** by entering your **email** and a **password**, or using a **Credential provider**. It will create your **account** on **Convertigo Cloud Trial**.



If you **create other workspaces**, you will just need to **log in to this account**.



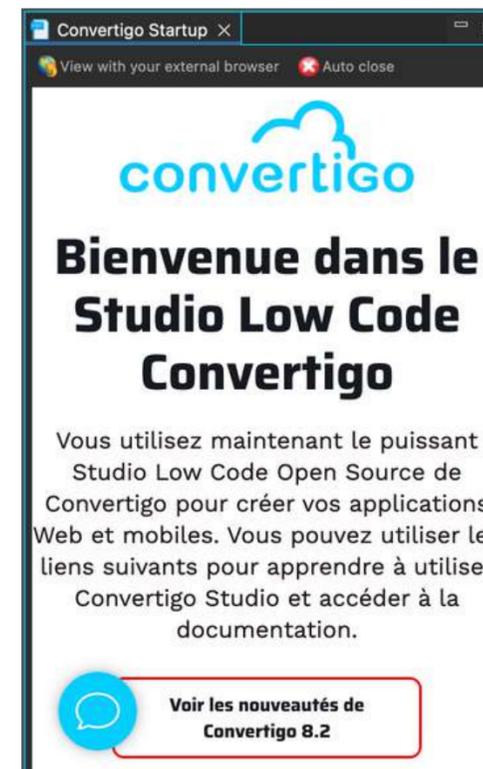
# 2.4 Configuration

## Step 6 – Welcome to Convertigo Low Code Studio

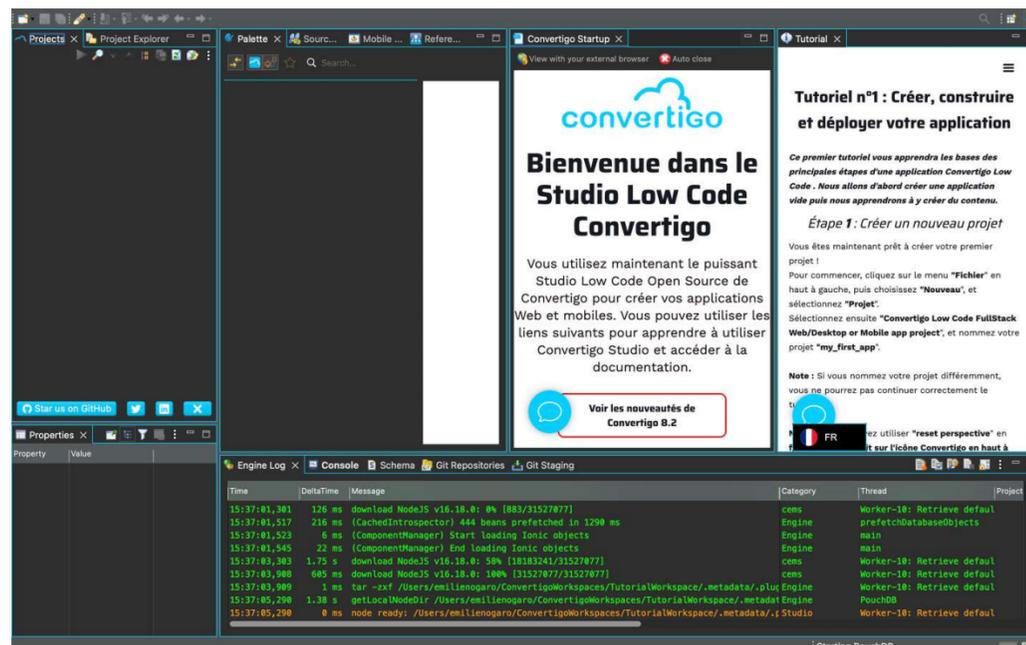


On first launch, you will find 2 additional views :

The **Convertigo Startup view** with a link to **Convertigo's website** and the studio's **documentation**.



The **Tutorial view** featuring **exercises** to help you **getting started** with the studio.

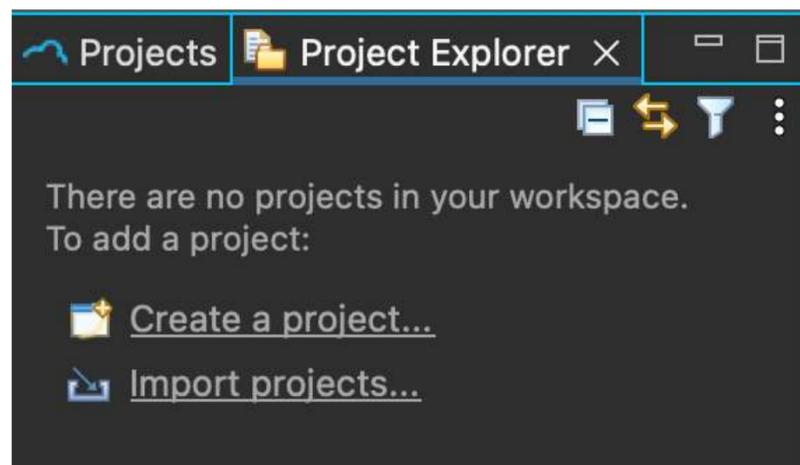


## 2.5 Create a project

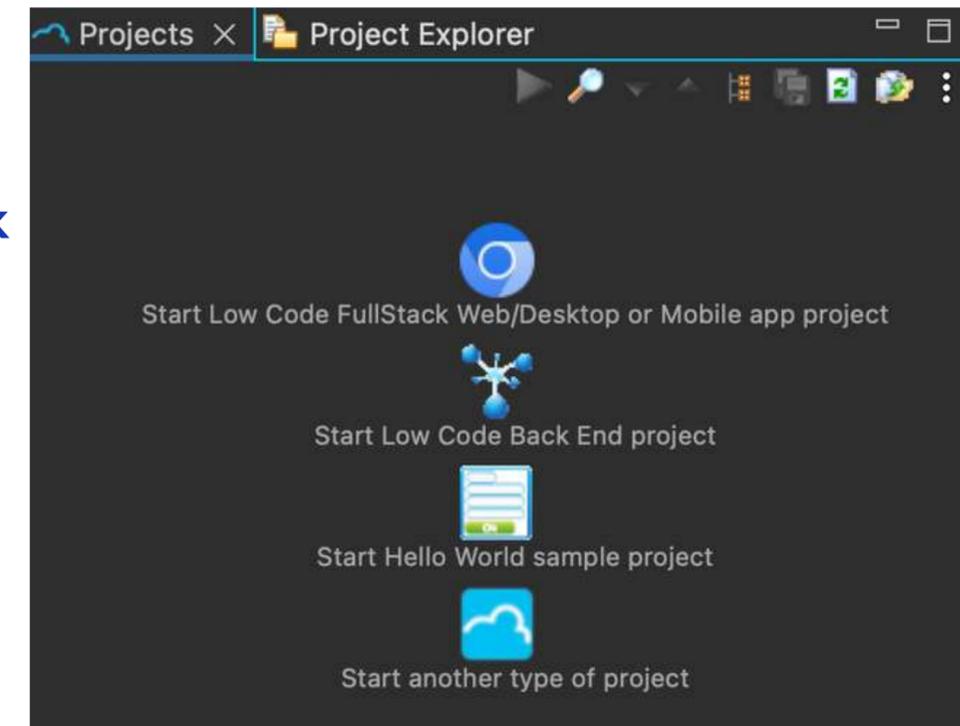
There are several ways to create a project in Convertigo.

When you create a project **for the first time** in a **new workspace**, you can :

First option :  
click on **Create a project**  
in the **Project Explorer view**



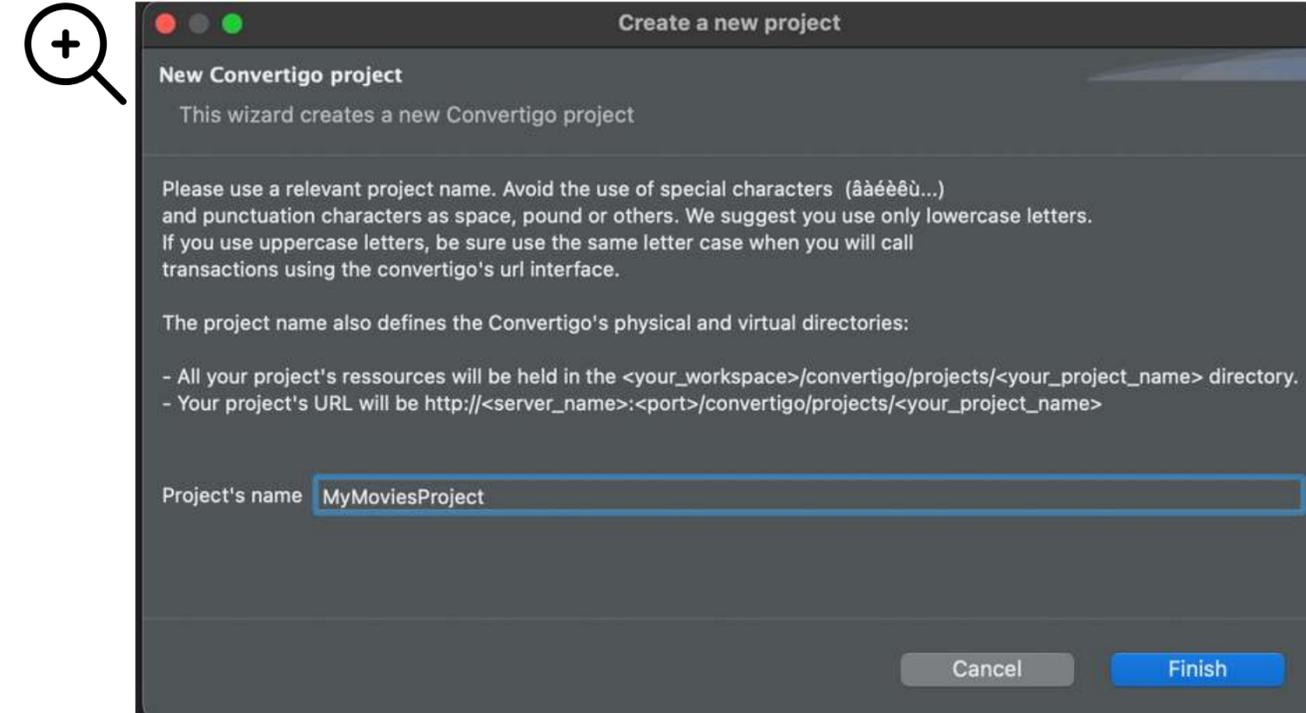
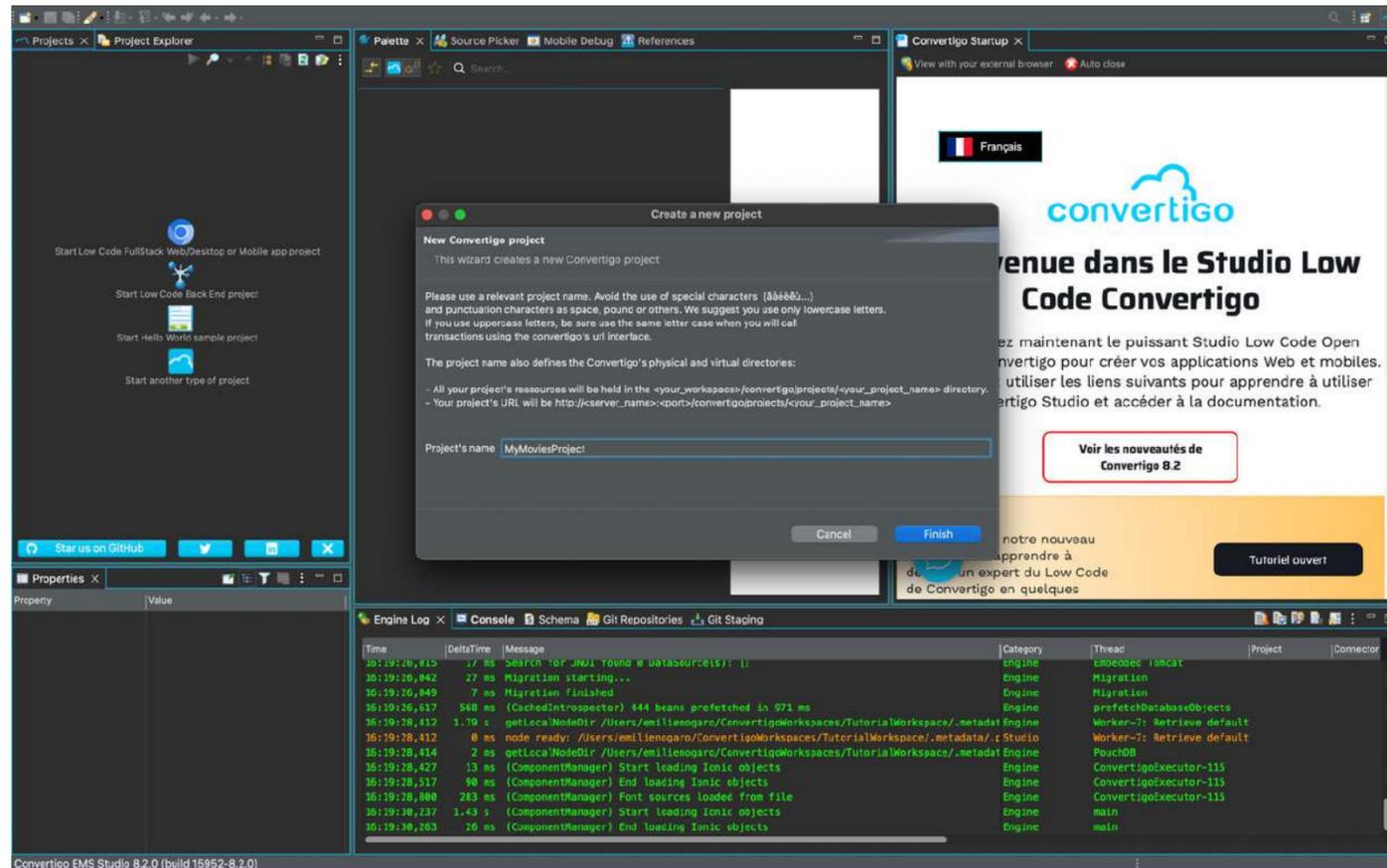
Second option :  
click on  
**Start Low Code Fullstack  
Web/Desktop  
or Mobile app project**  
in the **Project view**



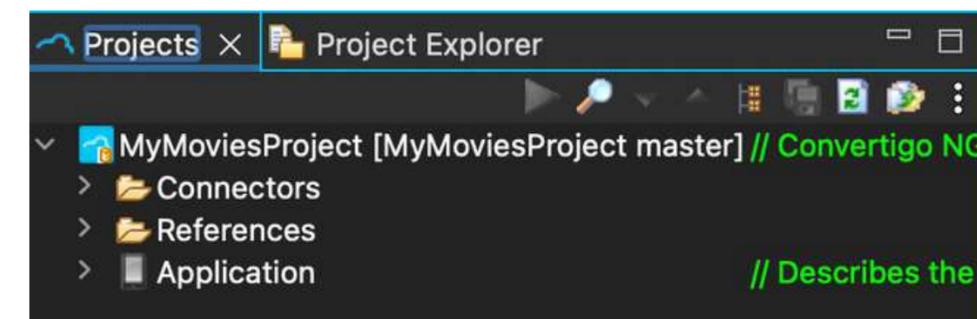
# 2.5 Create a project

The Create a new project windows appears.

Enter a project name, then click on **Finish**.

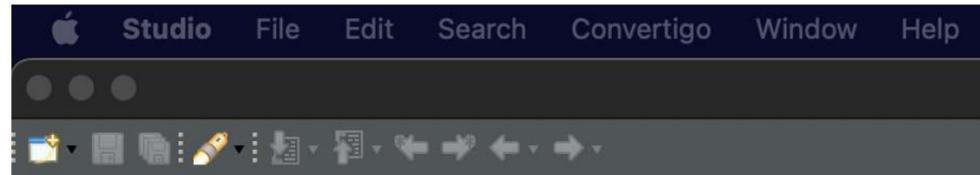


The project appears in the **Project view**.



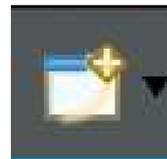
# 2.5 Create a project

Another way to create a project is to use the toolbar in the **Project view**.



## First option :

Click directly on this icon.



Click on **New>**  
then select **Project**.

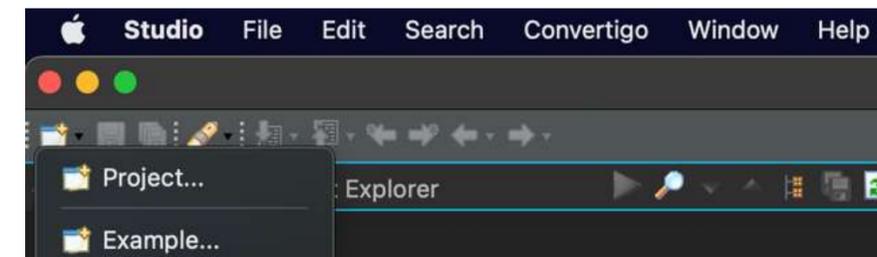


## Second option

Click on the arrow on the right of this icon.



A sub menu appears.  
Click on **Project** in the menu.

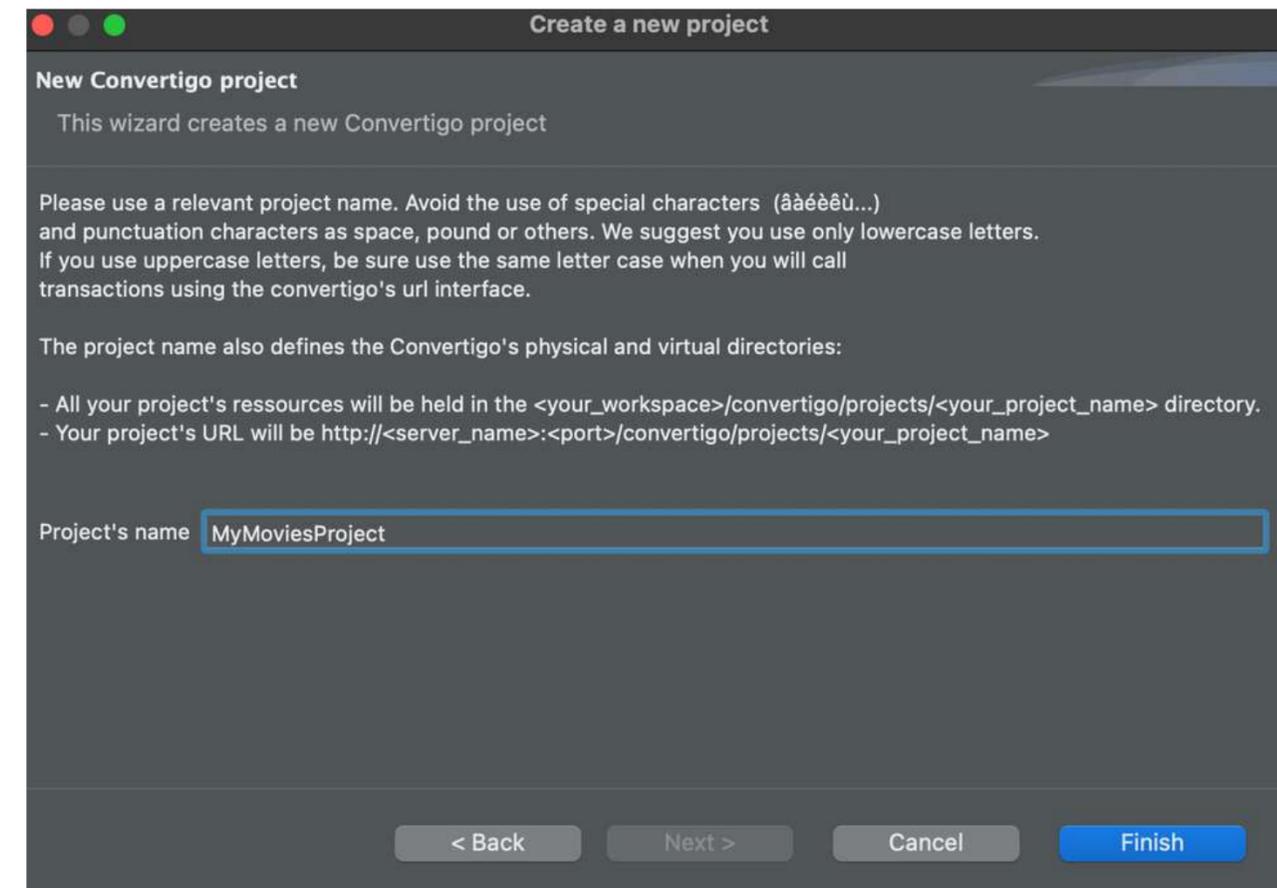
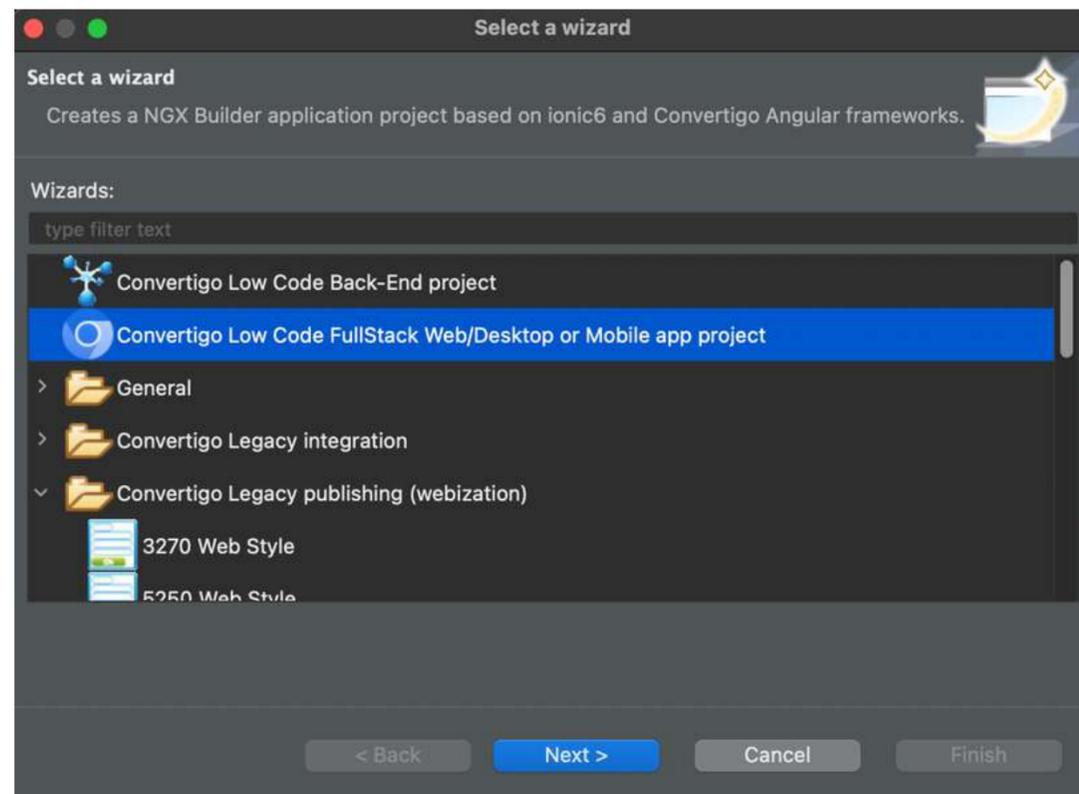


## 2.5 Create a project

Both options open the **Select a wizard window**.

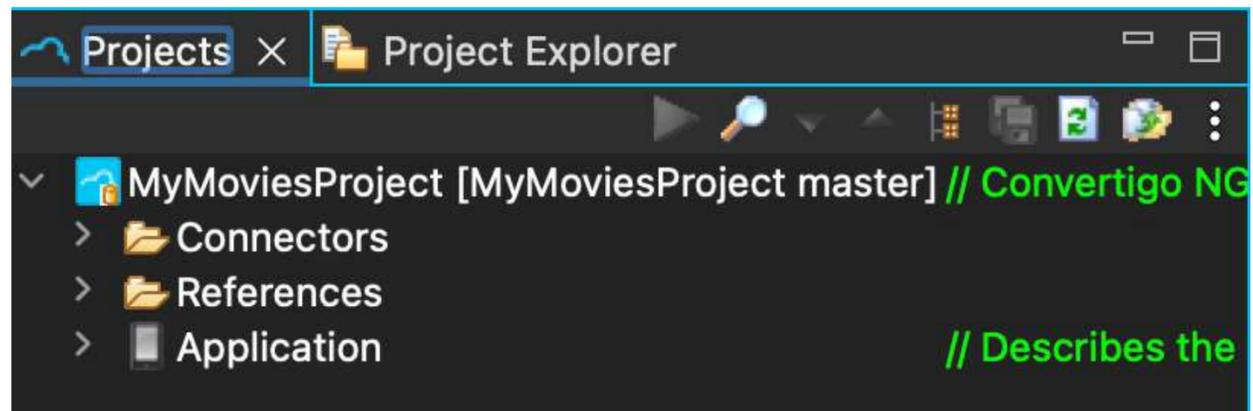
Select **Convertigo Low Code FullStack Web/Desktop or Mobile app project** and click on **Next>**.

Enter a project name, then click on **Finish**.

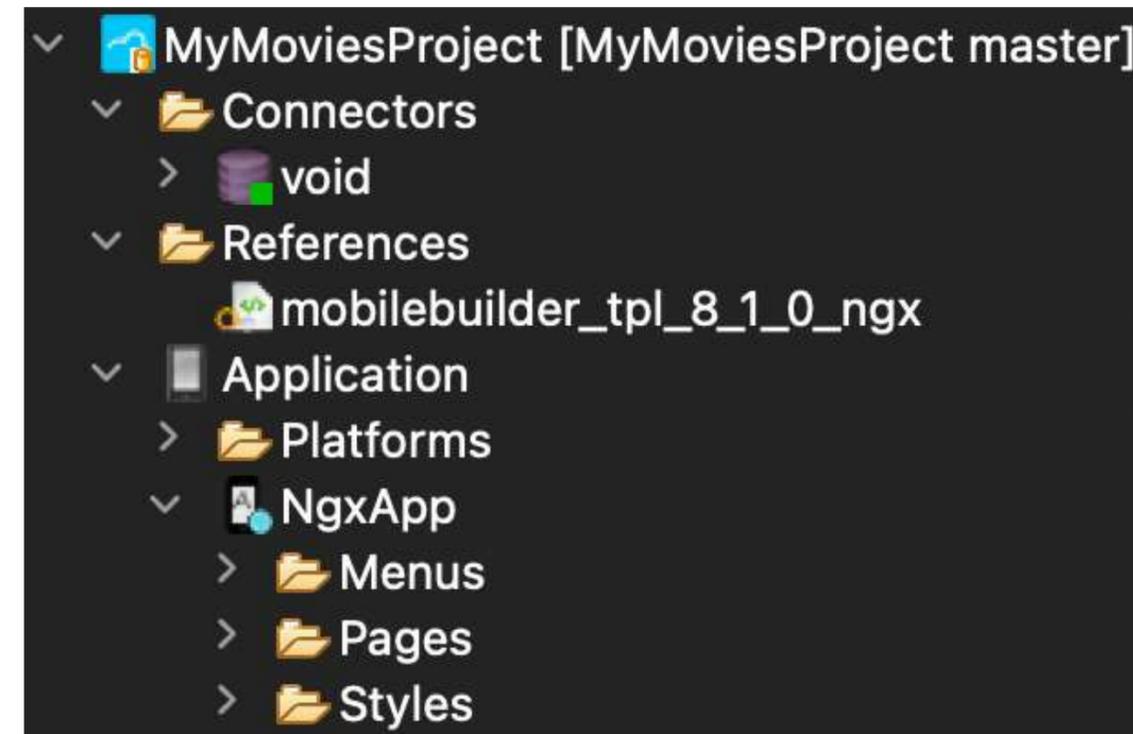


## 2.5 Create a project

As seen before, the project is created and appears in the **Project view**.



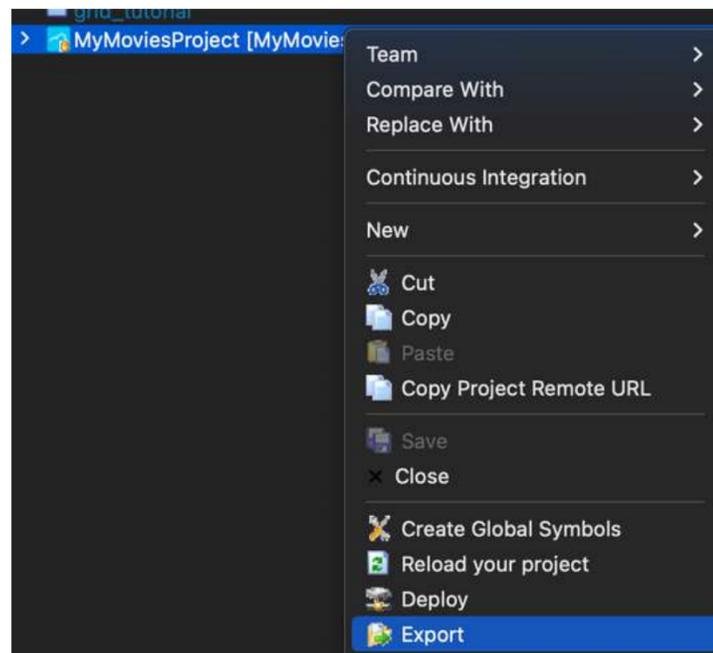
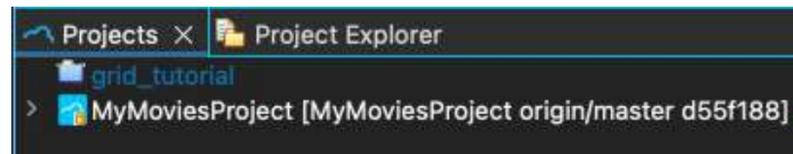
When created, a project has always the same structure.



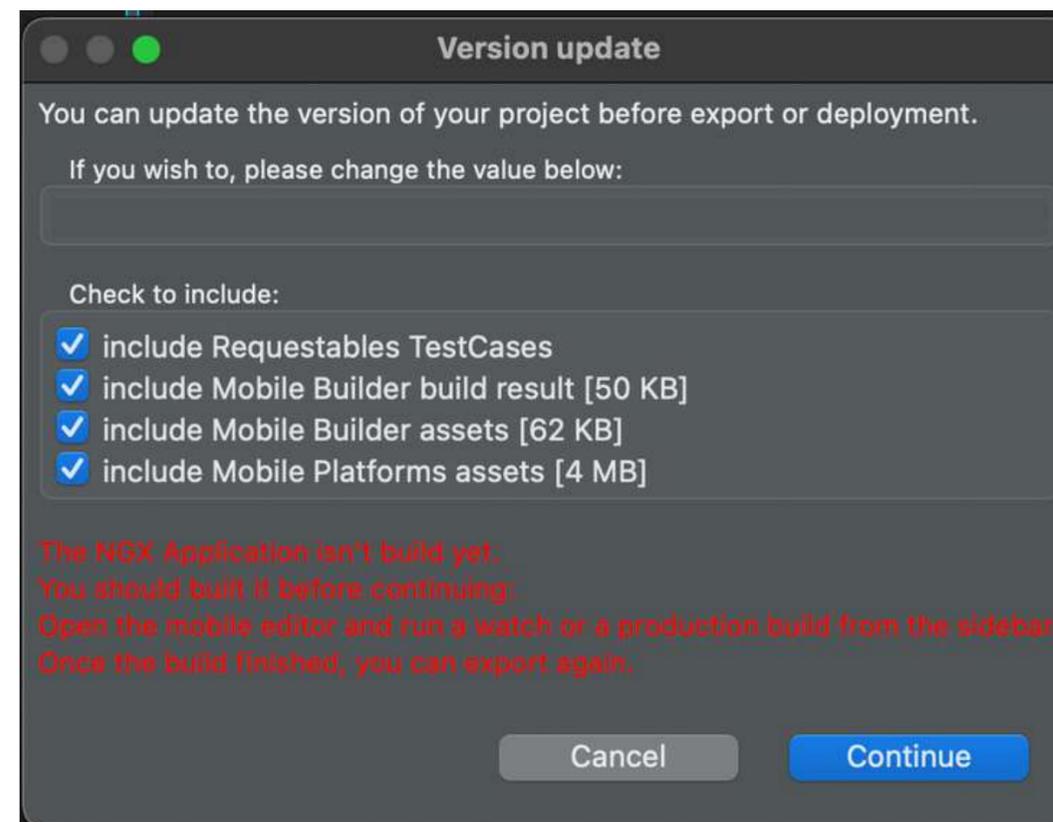
## 2.6 Export a project

Let's say you want to export a project:

right-click on the **name of your project** in the **Projects view**,  
then click on **Export**.



The **Version update window** appears.



A message in red indicates that you need to run a **watch** or a **production build** from the **mobile editor**.

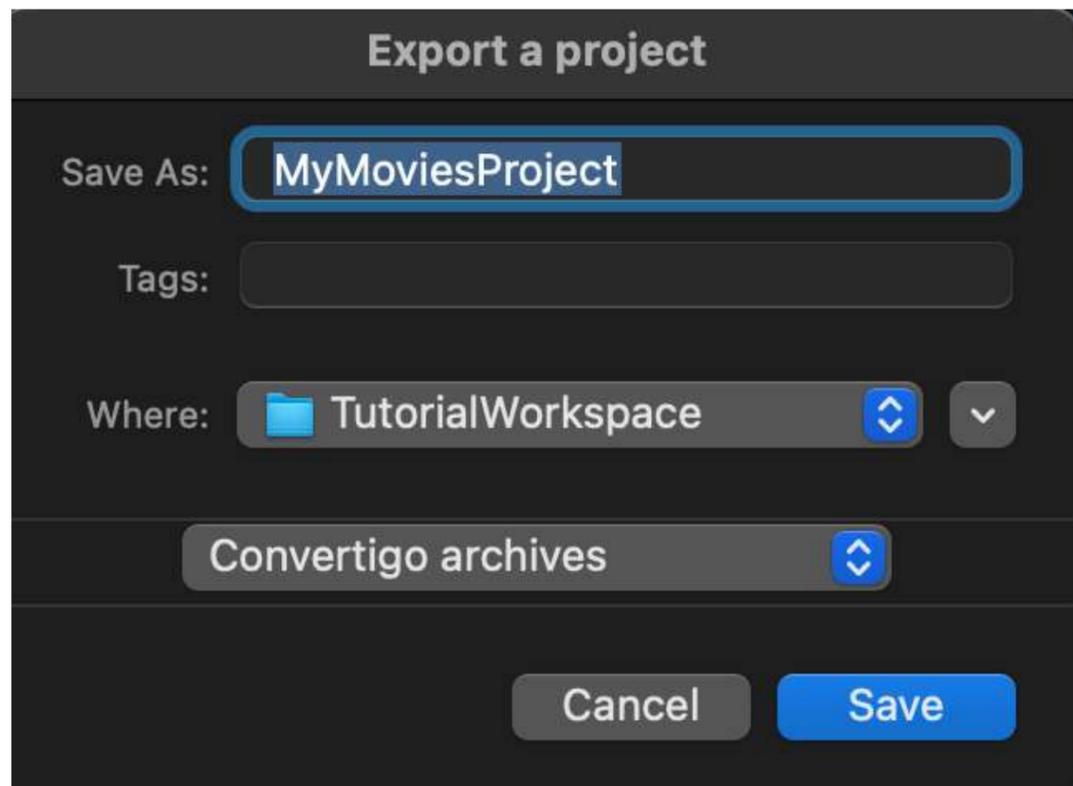
**Building the project** is necessary **only for the frontend**.

For now, we are **working on the backend**, so we can ignore this message, and click on **Continue**.

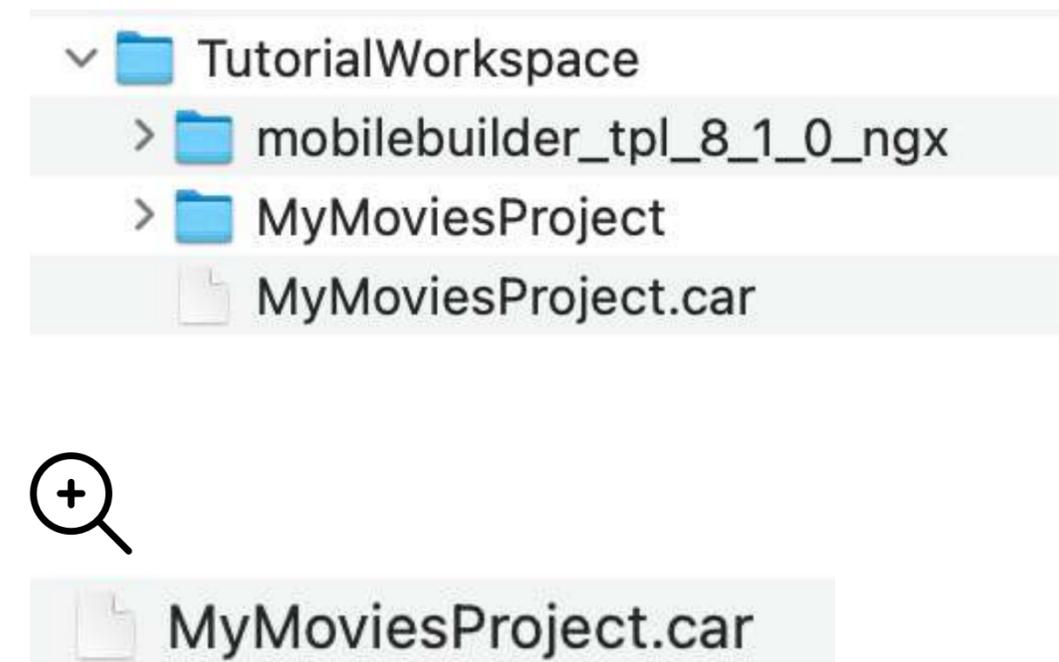


## 2.6 Export a project

In the **Export a project window**, you can change the name of the project, and select where it is saved.

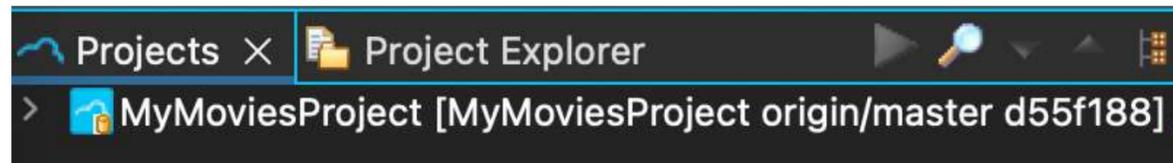


In the folder where it was saved, the project appears as a **.car file**.

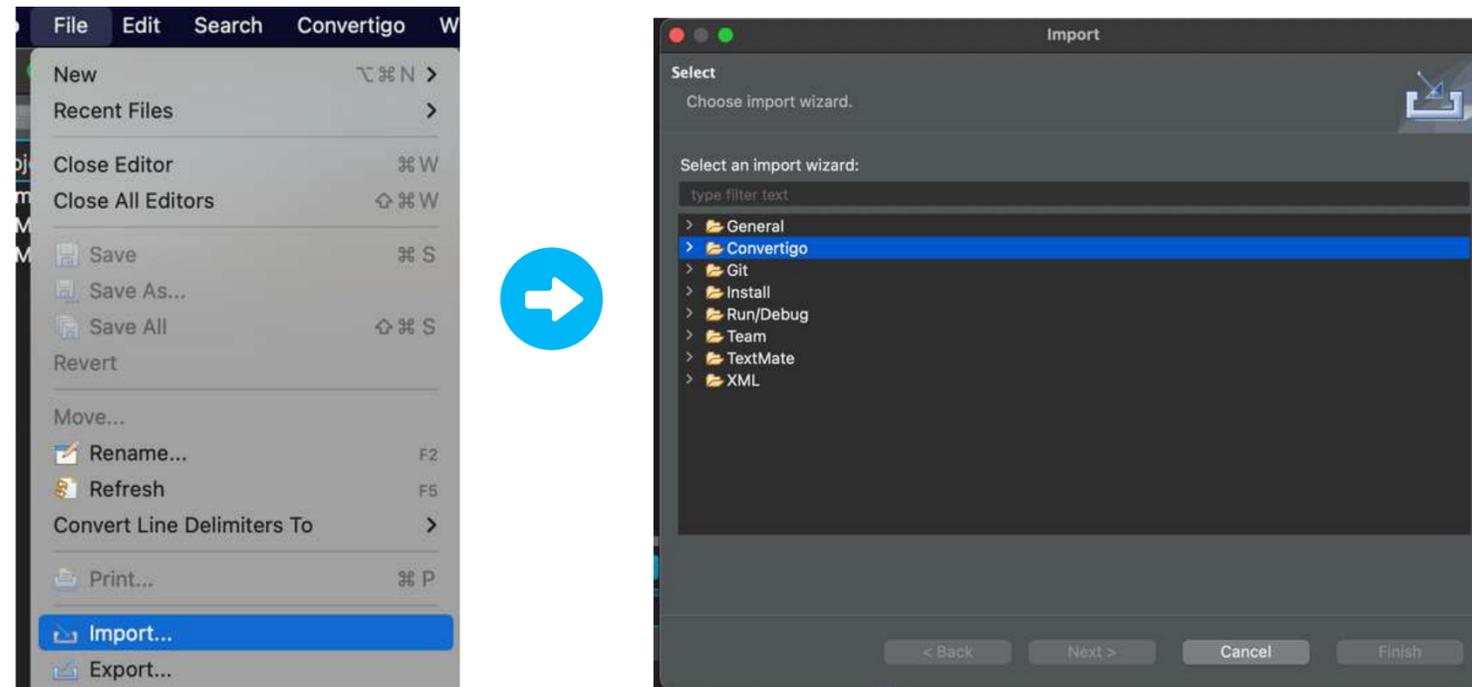


## 2.7 Import a project

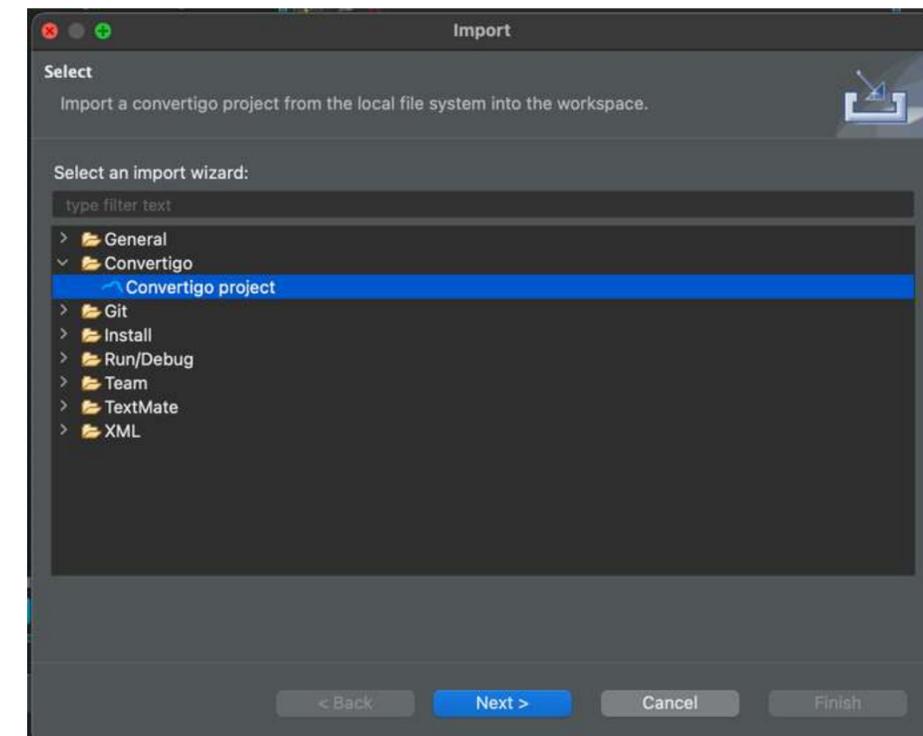
Let's say you want to import a project from a .car file in your workspace



Click on **File**, then **Import** and the **Import windows** appears.



In the **Import windows**, click on **Convertigo**, select **Convertigo project**, then click on **Next >**.

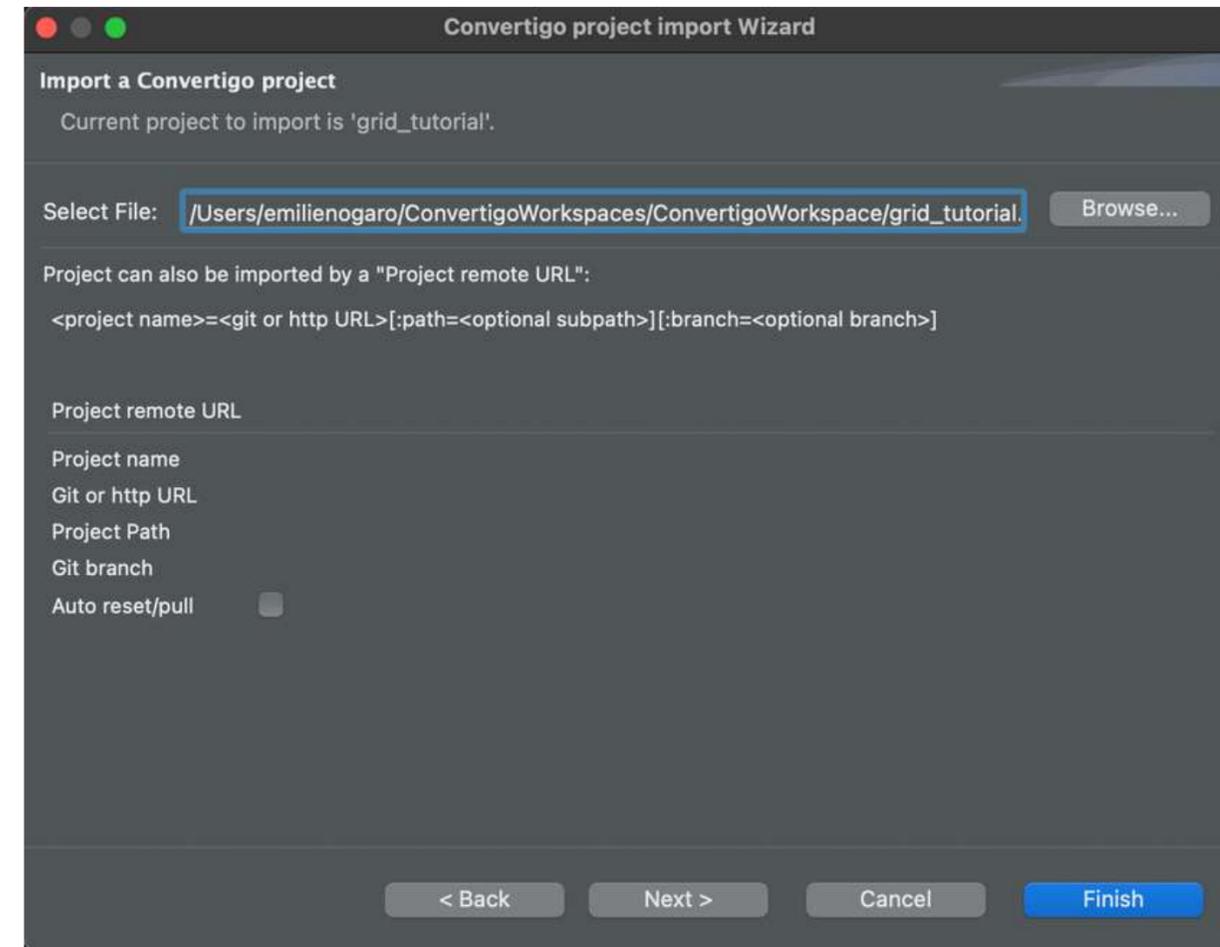
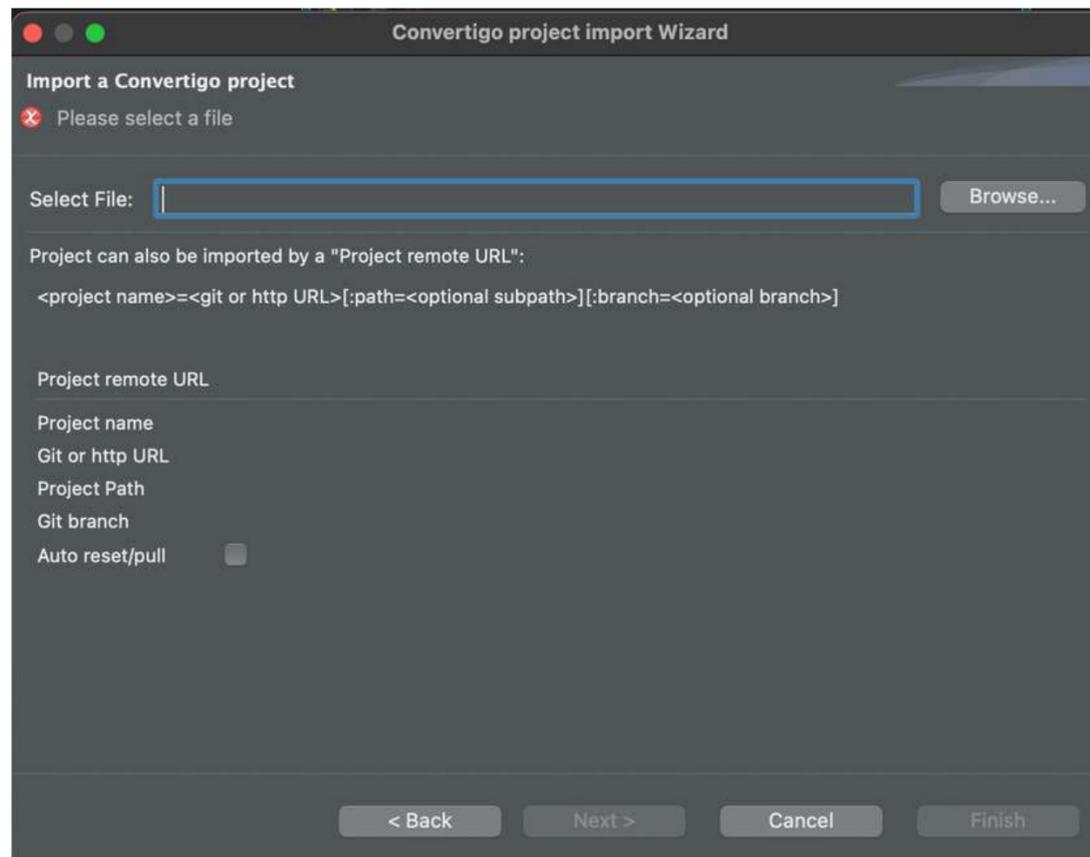


## 2.7 Import a project

In the **Convertigo Project Import** window, click on **Browse** to select a file (here `grid_tutorial.car`) anywhere in your computer.



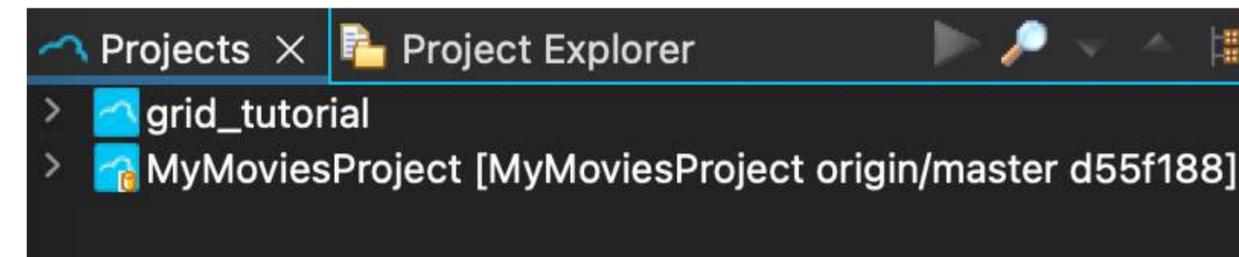
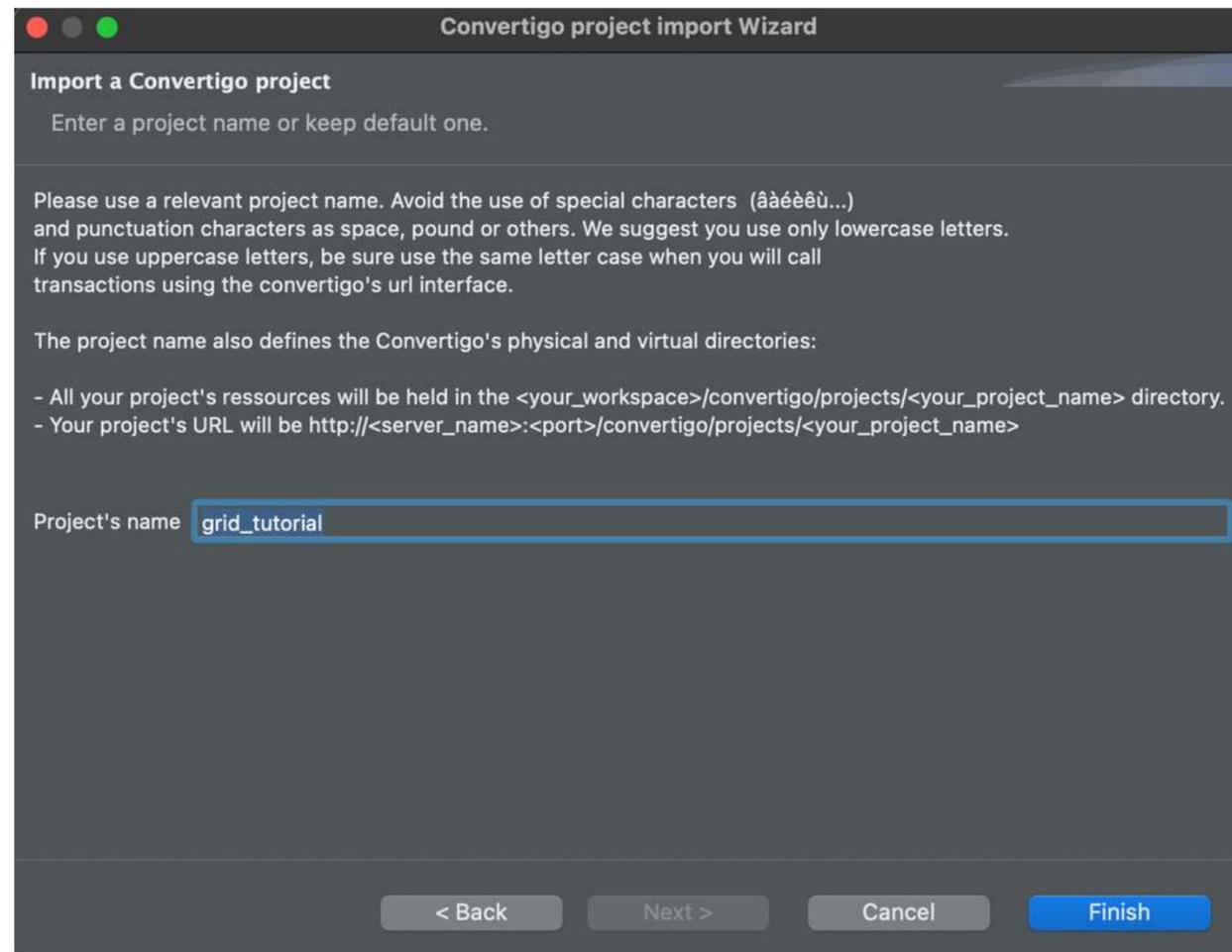
Then click on **Next>**.



## 2.7 Import a project

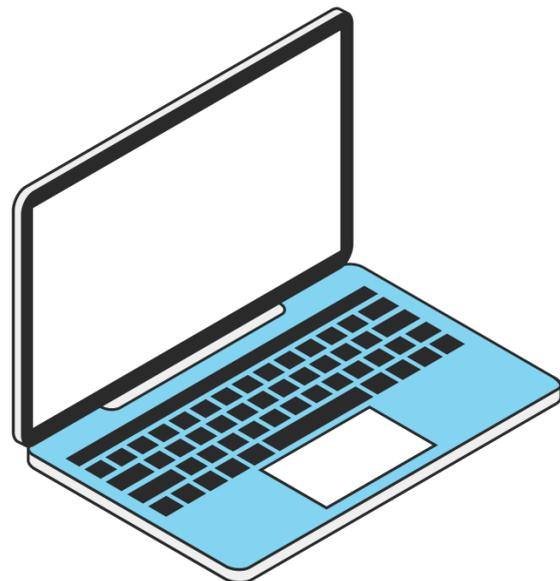
You can rename the project  
or keep the .default file name (.car file name).  
Then click on **Finish**.

In the **Projects view**, the project appears.



# 3 – Web services Connectors & Transactions

How to consume a rest API.



- 3.1 Presentation of the API TMDB

---
- 3.2 HTTP connectors & JSON HTTP transactions

---
- 3.3 Create an HTTP connector

---
- 3.4 Configure the HTTP connector

---
- 3.5 Create a transaction

---
- 3.6 Add a token

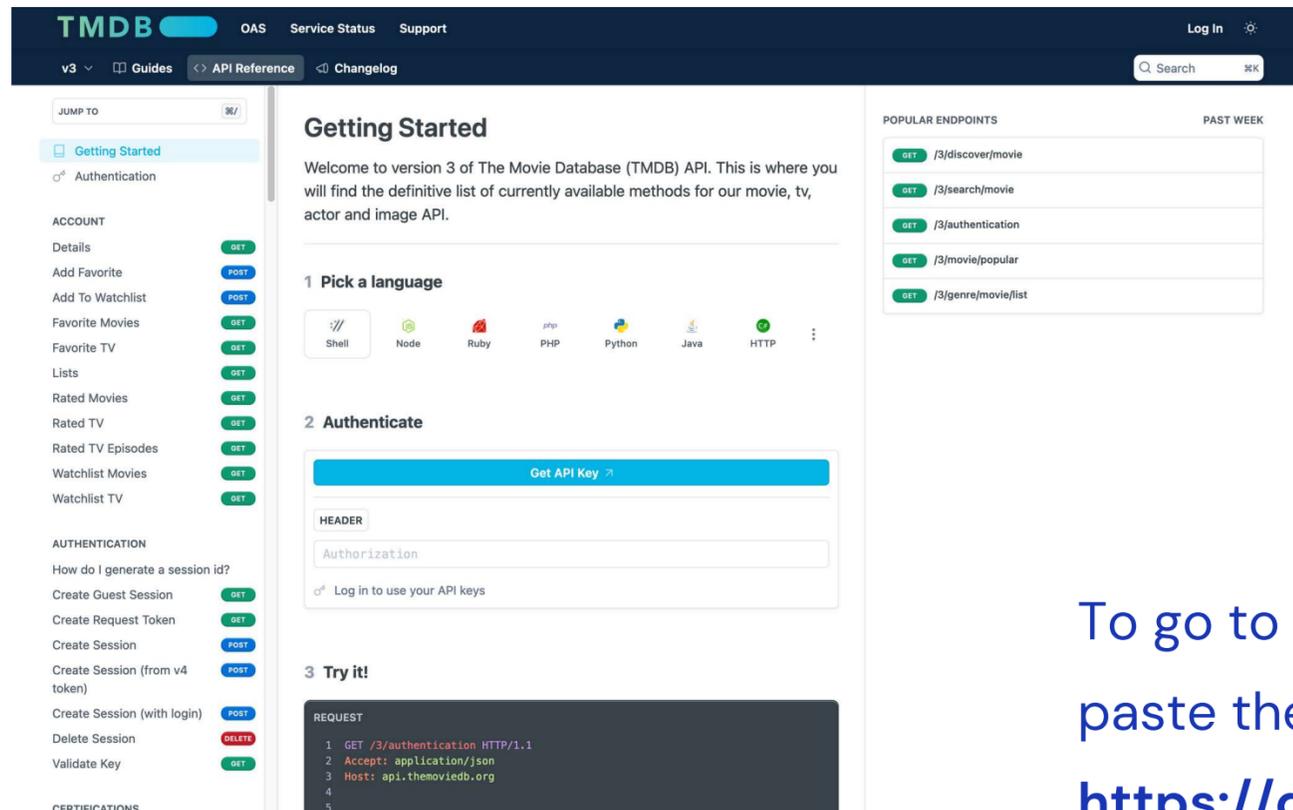
---
- 3.7 Edit the request path

---
- 3.8 Test the request

# 3.1 Presentation of the API TMDb

The Movie Database (TMDb) API provides access to a vast database of information related to movies and television shows.

It is commonly used by developers to integrate movie-related data into their applications, websites, and services

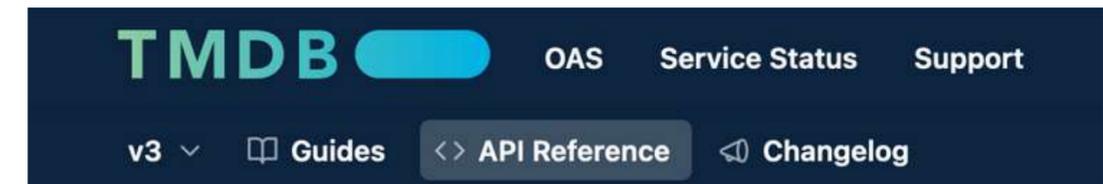


The screenshot shows the TMDb API documentation page for version 3. The main content area is titled "Getting Started" and contains the following sections:

- Welcome:** Welcome to version 3 of The Movie Database (TMDb) API. This is where you will find the definitive list of currently available methods for our movie, tv, actor and image API.
- 1 Pick a language:** A list of programming languages and protocols with their respective icons: Shell, Node, Ruby, PHP, Python, Java, and HTTP.
- 2 Authenticate:** A section with a "Get API Key" button and a "HEADER" field containing "Authorization". Below it, a link says "Log in to use your API keys".
- 3 Try it!:** A "REQUEST" section showing a sample request:

```
1 GET /3/authentication HTTP/1.1
2 Accept: application/json
3 Host: api.themoviedb.org
4
5
```

The left sidebar contains a navigation menu with categories like "Getting Started", "Authentication", "ACCOUNT", "AUTHENTICATION", and "CERTIFICATIONS". The top navigation bar includes "v3", "Guides", "API Reference", "Changelog", "Log In", and "Search".



The screenshot shows the top navigation bar of the TMDb API documentation. It features the "TMDb" logo, "OAS", "Service Status", and "Support" links. Below the logo, there are "v3", "Guides", "API Reference", and "Changelog" links.



## Getting Started

Welcome to version 3 of The Movie Database (TMDb) API. This is where you will find the definitive list of currently available methods for our movie, tv, actor and image API.

To go to the **Getting started page** of the API, paste the following link in your browser:

<https://developer.themoviedb.org/reference/intro/getting-started>



# 3.1 Presentation of the API TMDb



In the API TMDb documentation, a lot of different requests are available.

Let's go to the **Search Movie page** (<https://developer.themoviedb.org/reference/search-movie>).

**SEARCH**

- Collection GET
- Company GET
- Keyword GET
- Movie GET**
- Multi GET
- Person GET
- TV GET

### Movie

GET <https://api.themoviedb.org/3/search/movie>

Search for movies by their original, translated and alternative titles.

LOG IN TO SEE FULL REQUEST HISTORY

TIME	STATUS	USER AGENT
Make a request to see history.		
0 Requests This Month		SEE ALL REQUESTS →

QUERY PARAMS

- query string required
- include\_adult boolean false
- language string en-US
- primary\_release\_year string
- page int32 1
- region string
- year string

RESPONSE

- 200

LANGUAGE: Shell, Node, Ruby, PHP, HTTP

AUTHORIZATION: Header, Authorization

Log in to use your API keys

REQUEST

```
1 GET /3/search/movie?include_adult=false&language=en-US
2 Accept: application/json
3 Host: api.themoviedb.org
4
5
```

Try It!

RESPONSE

Click **Try It!** to start a request and see the response here! Or choose an example:

- application/json
- 200 - Result



# 3.1 Presentation of the API TMDB

All the informations you need to write a **Search Movie HTTP REQUEST** are present on the **Search Movie page**.

**GET HTTP Request url** to search a movie

## Movie

**GET** <https://api.themoviedb.org/3/search/movie>

Search for movies by their original, translated and alternative titles.

## Expected Response code

RESPONSE

● 200  
200



## Required and Optional Query params.

QUERY PARAMS

**query** string **required**

**include\_adult** boolean

**language** string

**primary\_release\_year** string

**page** int32

**region** string

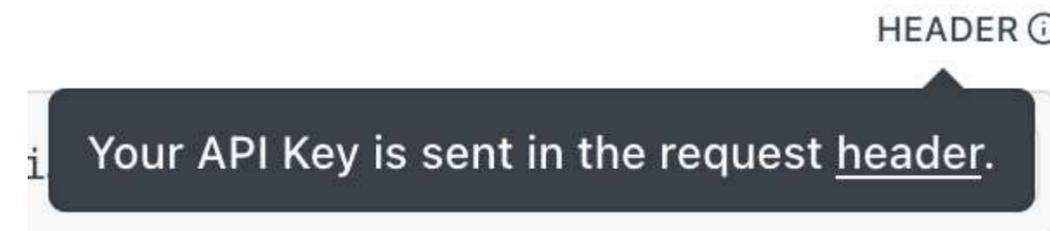
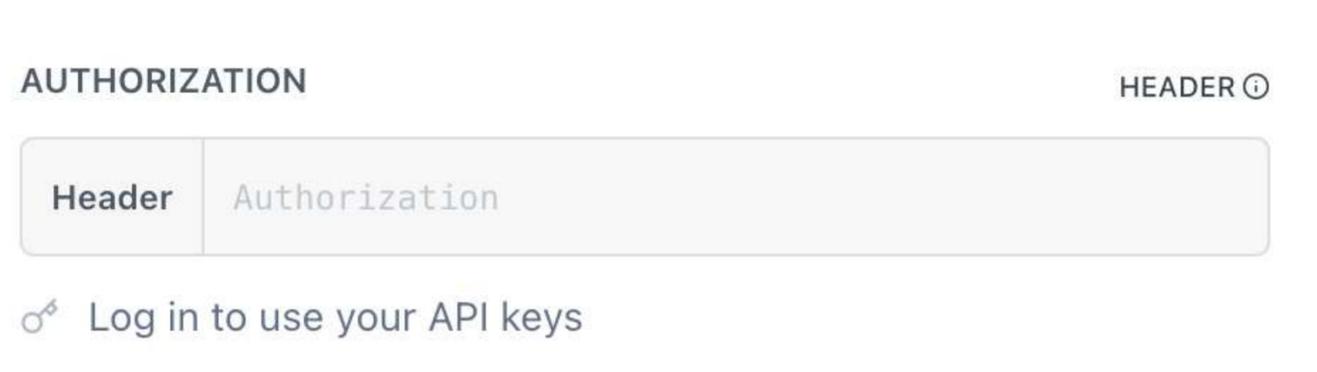
**year** string



# 3.1 Presentation of the API TMDB

To use the API TMDB, it is **necessary to create an account.**

Once registered, you will have an **API Key** or **personnal Access Token**.  
It will be used in the **request Header** as **Authorization param.**



The **personnal access token** appears **automatically** when you are logged in.



# 3.1 Presentation of the API TMDB

LANGUAGE

Shell Node Ruby PHP HTTP

↓

REQUEST

```
1 GET /3/search/movie?include_adult=false&language=en-US
2 Accept: application/json
3 Host: api.themoviedb.org
4
```

↓

QUERY PARAMS

query string required avatar

↓

REQUEST

```
1 GET /3/search/movie?query=avatar&include_adult=false&
2 Accept: application/json
3 Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOi
4 Host: api.themoviedb.org
5
```

Change the **Language** to **HTTP** to see the request as HTTP

Request url by default

GET /3/search/movie?include\_adult=false&language=en-US&page=1

When you add a query param,  
the **request url changes** to include it.

GET /3/search/movie?

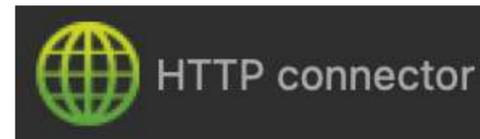
query=avatar&include\_adult=false&language=en-US&page=1



# 3.2 HTTP connectors & JSON HTTP transactions

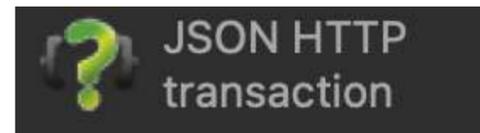


There are **different connectors and transactions** in Convertigo, used for **different data providers** (SQL, Web services, Legacy apps running on mainframes...).



For a REST API, you use the **HTTP connector**.

It allows Convertigo to connect and communicate with **HTTP servers**. It is used to **consume REST and SOAP web services**, and retrieve data using the **HTTP protocol**.



To consume a **JSON web service**, you use a **JSON HTTP transaction**.

It performs the conversion of JSON data from the web service into XML transaction output.

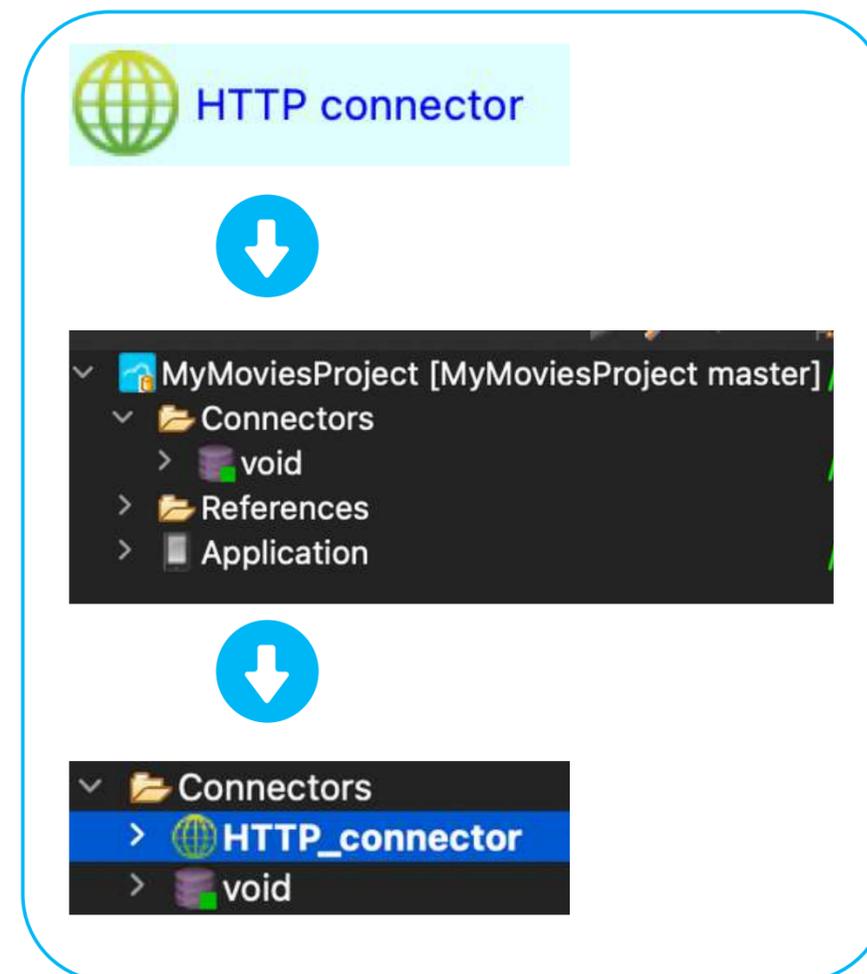


## 3.3 Create an HTTP connector

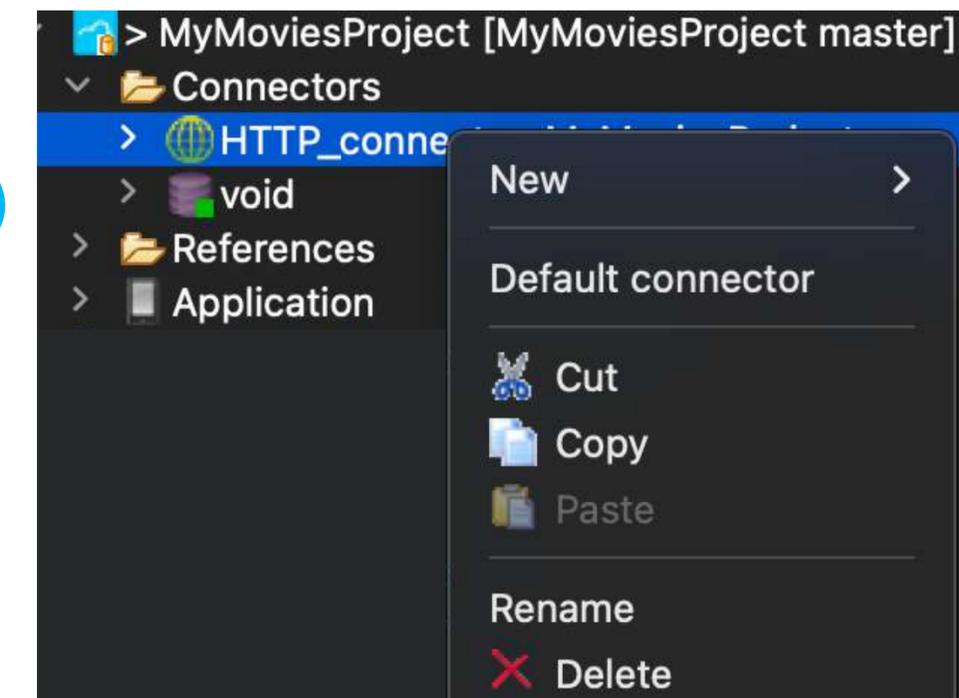
To connect to a **REST API**,  
you need to **create an HTTP\_connector** in the **Connectors** folder.

First option:

**Drag and drop** it from the palette into the folder.



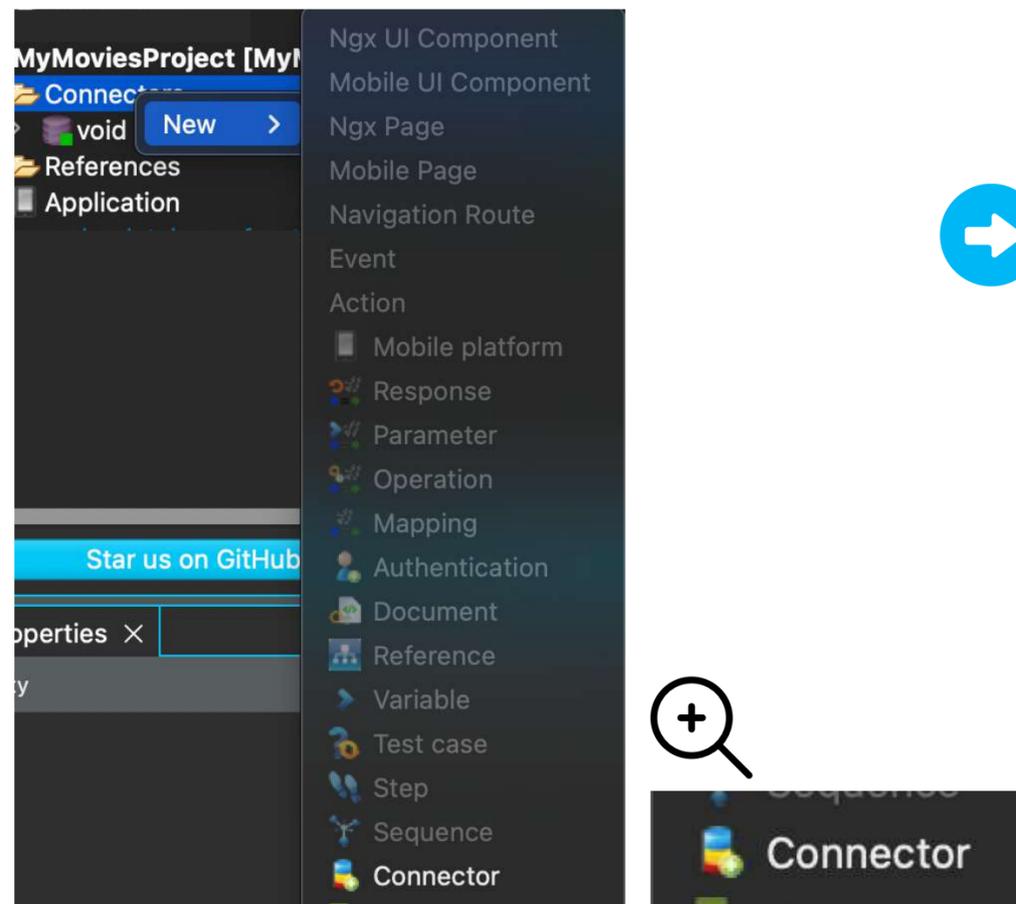
You can then **rename** the connector  
by right-clicking on it.



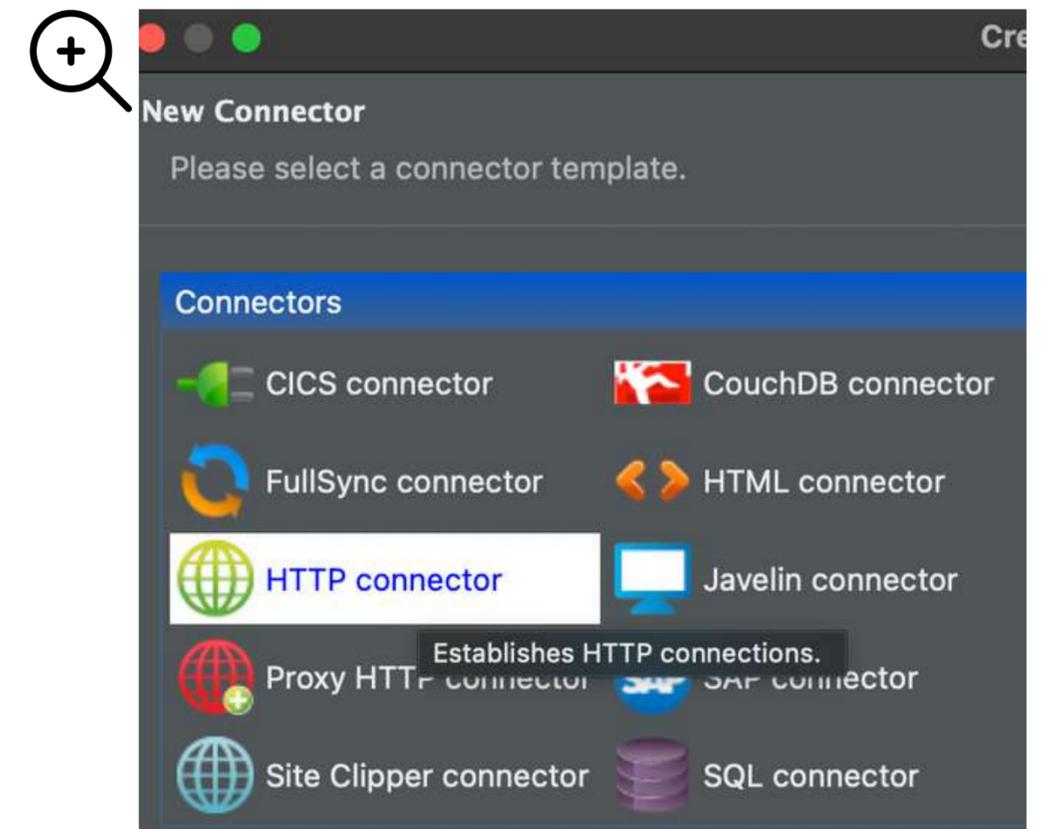
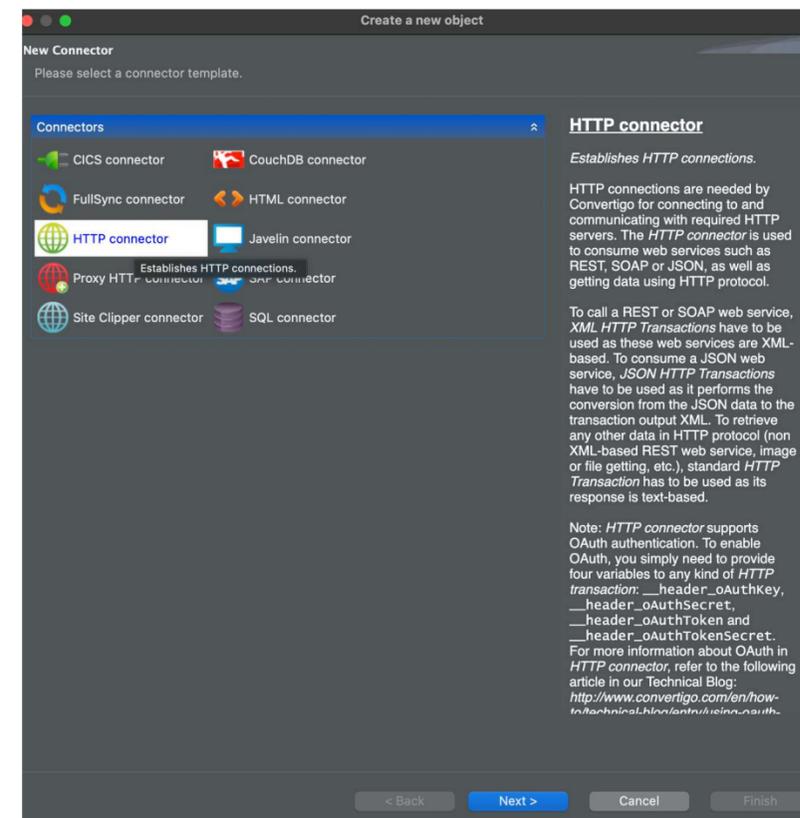
## 3.3 Create an HTTP connector

Second option:

Right-click on the **Connectors** folder, then select **New** and choose **Connector**.

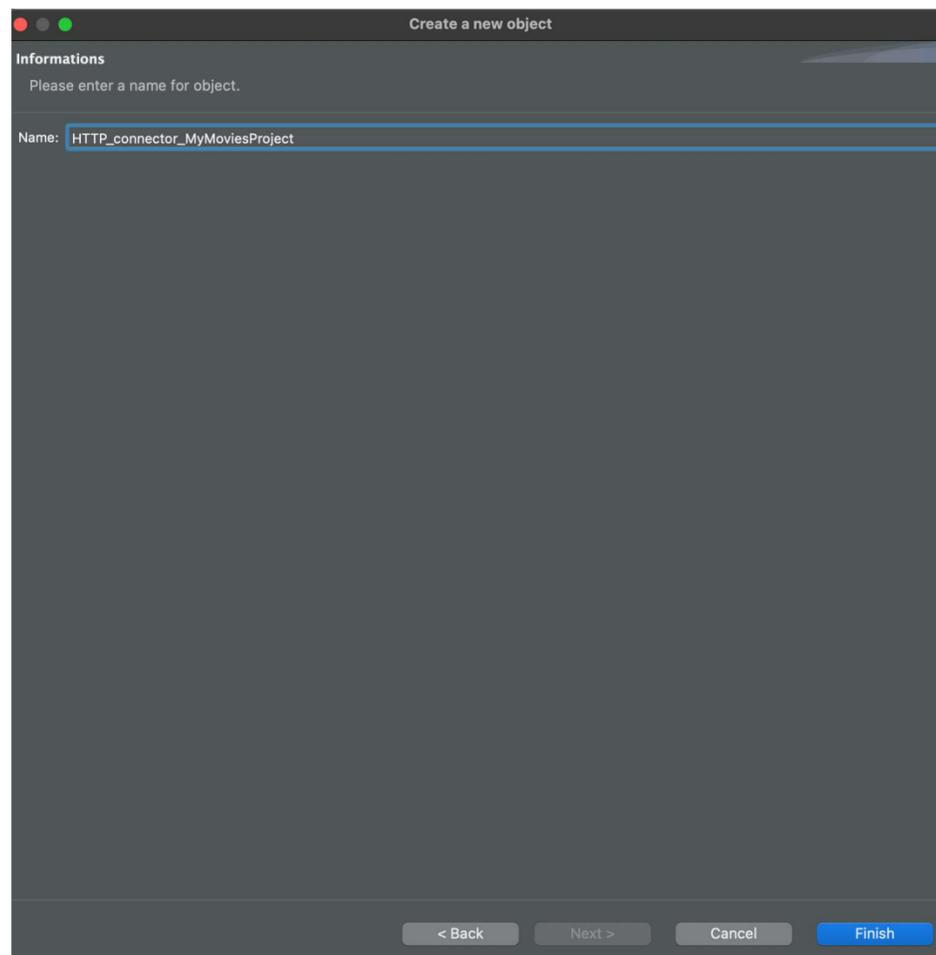


In the **Create a new object** window, select **HTTP connector** and then click on **Next >**.



## 3.3 Create an HTTP connector

Choose a name for the connector, and click on **Finish**.



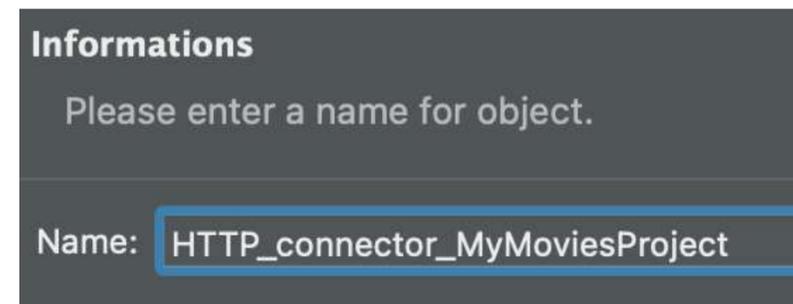
Create a new object

Informations

Please enter a name for object.

Name: HTTP\_connector\_MyMoviesProject

< Back Next > Cancel Finish



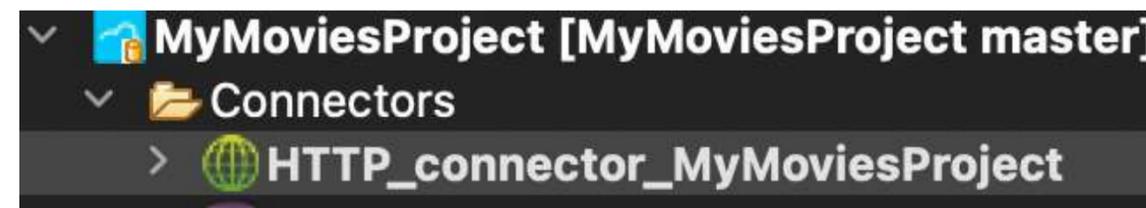
Informations

Please enter a name for object.

Name: HTTP\_connector\_MyMoviesProject

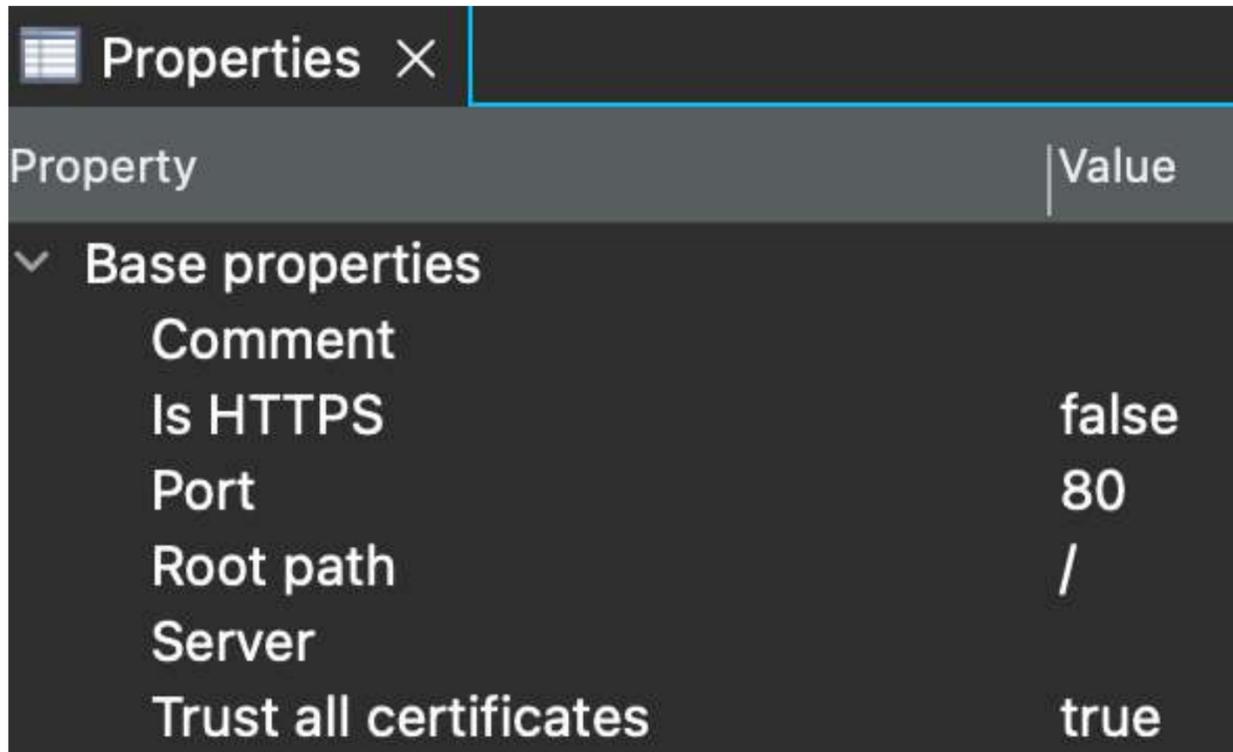


The new connector is created in the **Connectors** folder.



## 3.4 Configure the HTTP connector

In the **Properties window**,  
you will find the **default properties** of the connector.



Property	Value
Base properties	
Comment	
Is HTTPS	false
Port	80
Root path	/
Server	
Trust all certificates	true

For **http requests**

- IsHTTPS : false
- Port : 80

For **https requests**

- IsHTTPS : true
- Port : 443

Root path : / (default path)

Server : => enter a server name



## 3.4 Configure the HTTP connector

Now, we need to configure the connector with the informations found in the TMDb API documentation.

`GET https://api.themoviedb.org/3/search/movie`

```
REQUEST
1 GET /3/search/movie?include_ad
2 Accept: application/json
3 Host: api.themoviedb.org
```

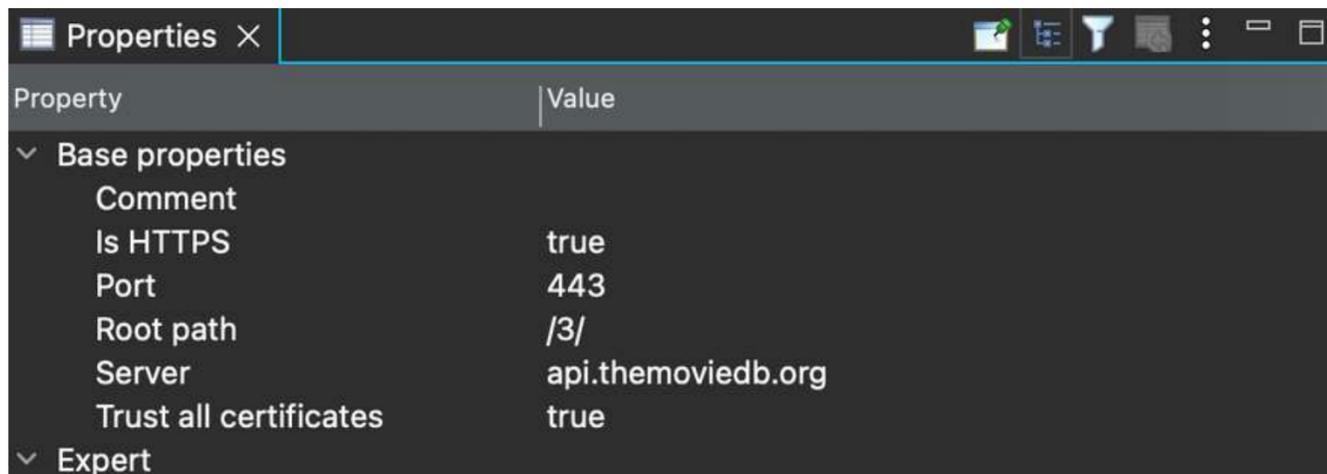


In the TMDb API documentation, we can see that:

- the request is **https**
- the request has a root path : **/3/** (version 3 of the API)
- the domain name is **api.themoviedb.org**

As a result, the **Connector configuration** is

- IsHTTPS : true
- Port : 443
- Root path : /3/
- Server : api.themoviedb.org

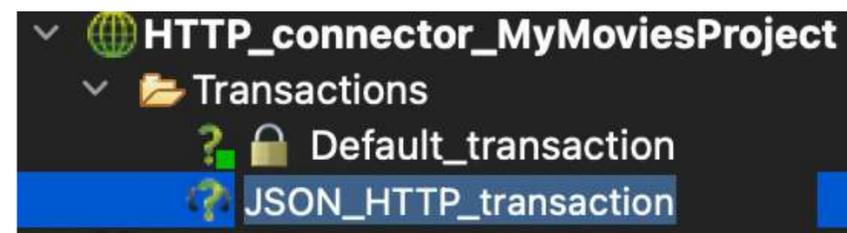
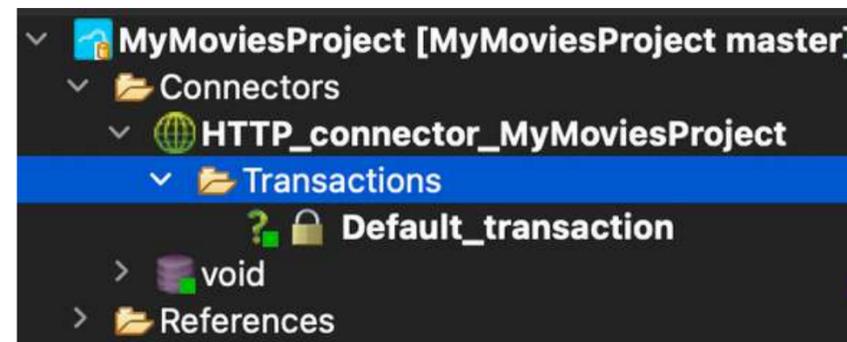
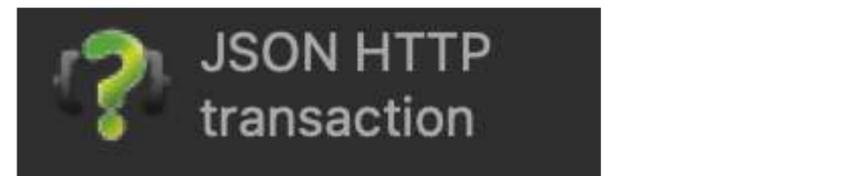
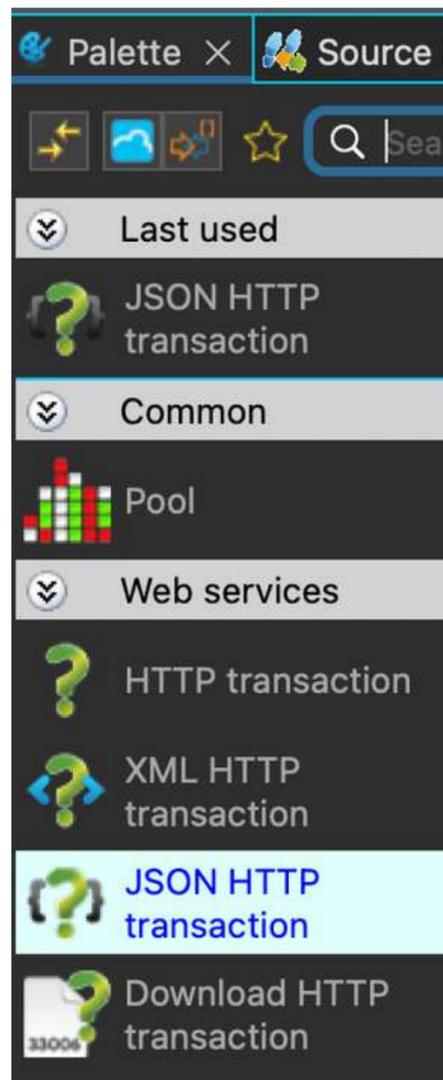


Property	Value
Base properties	
Comment	
Is HTTPS	true
Port	443
Root path	/3/
Server	api.themoviedb.org
Trust all certificates	true
Expert	

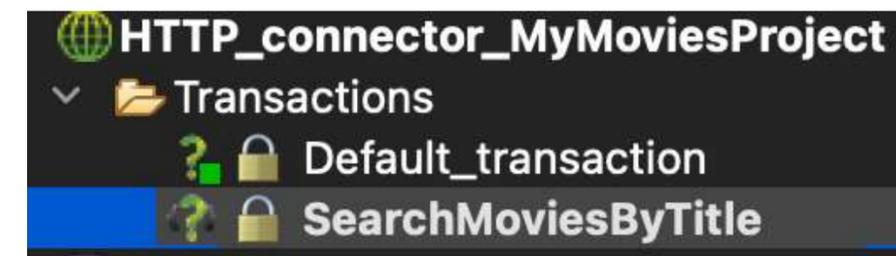


## 3.5 Create a transaction

First option : Drag and drop a **JSON HTTP transaction** from the palette into the **Connectors folder**.

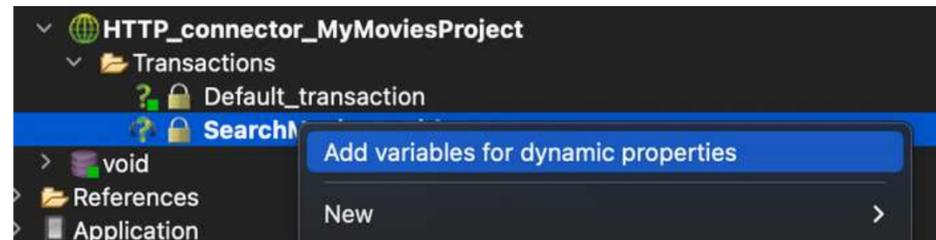


Rename the transaction to **SearchMoviesByTitle**.

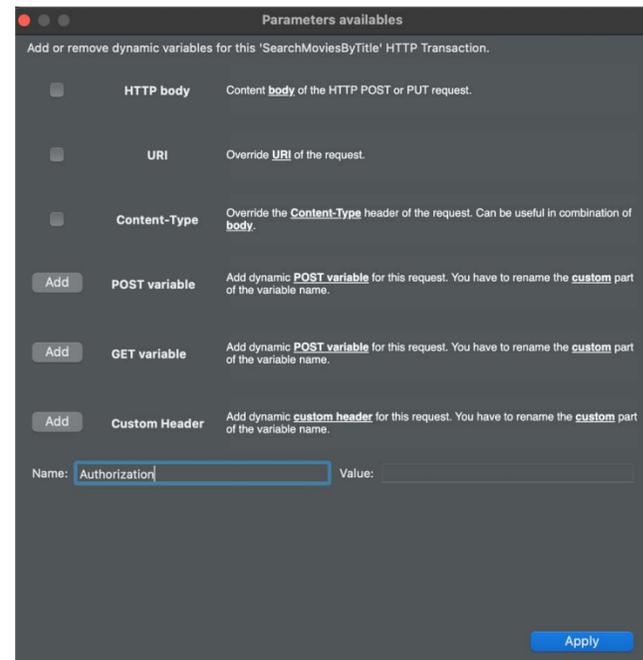


# 3.5 Create a transaction

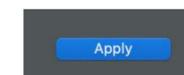
Right-click on the transaction, and select **Add variables for dynamic properties.**



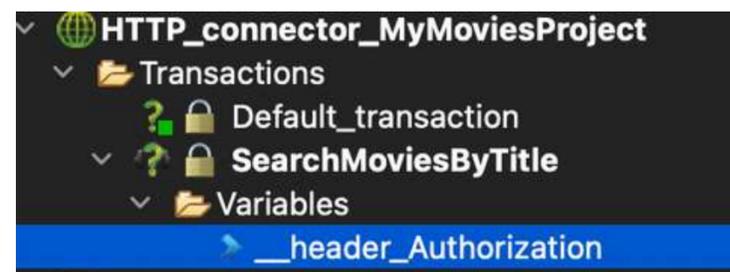
The **Parameters available** window appears.



In the **Parameters available** window, click on **Add Custom Header** to add the **Authorization** header (which will allow sending the access token),



Then click on **Apply**.



The **Authorization** header will appear in the folder **Variables** of the transaction

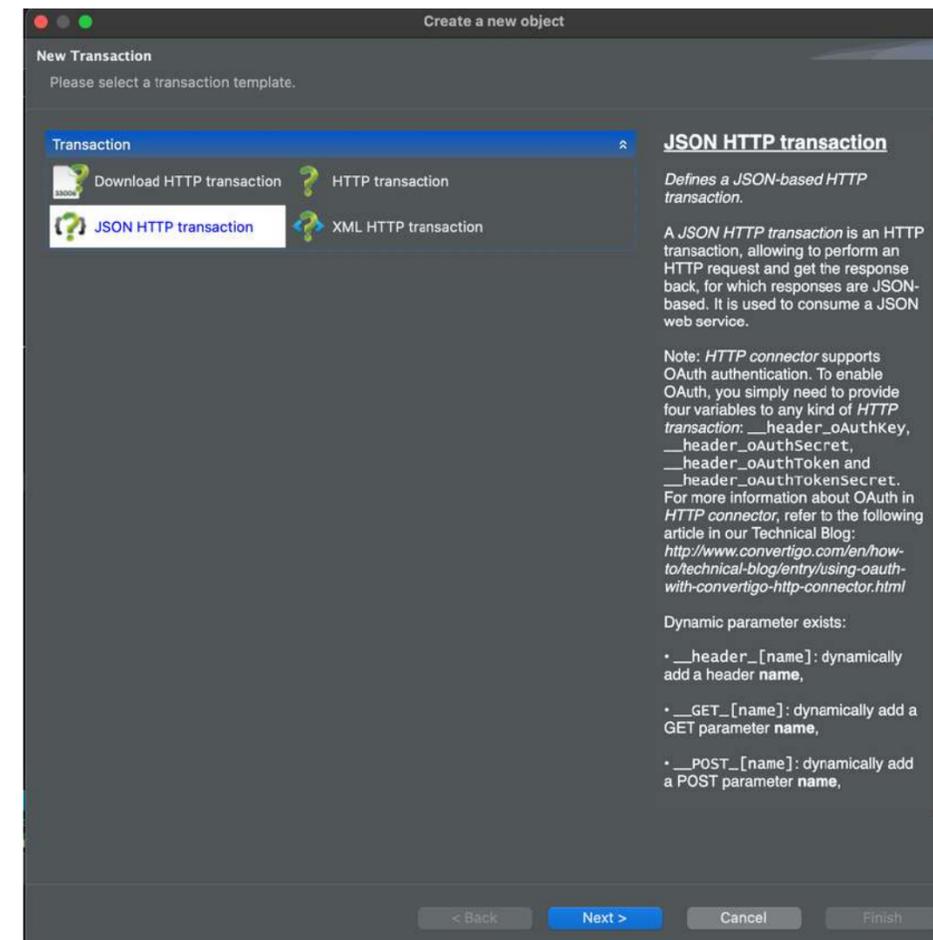
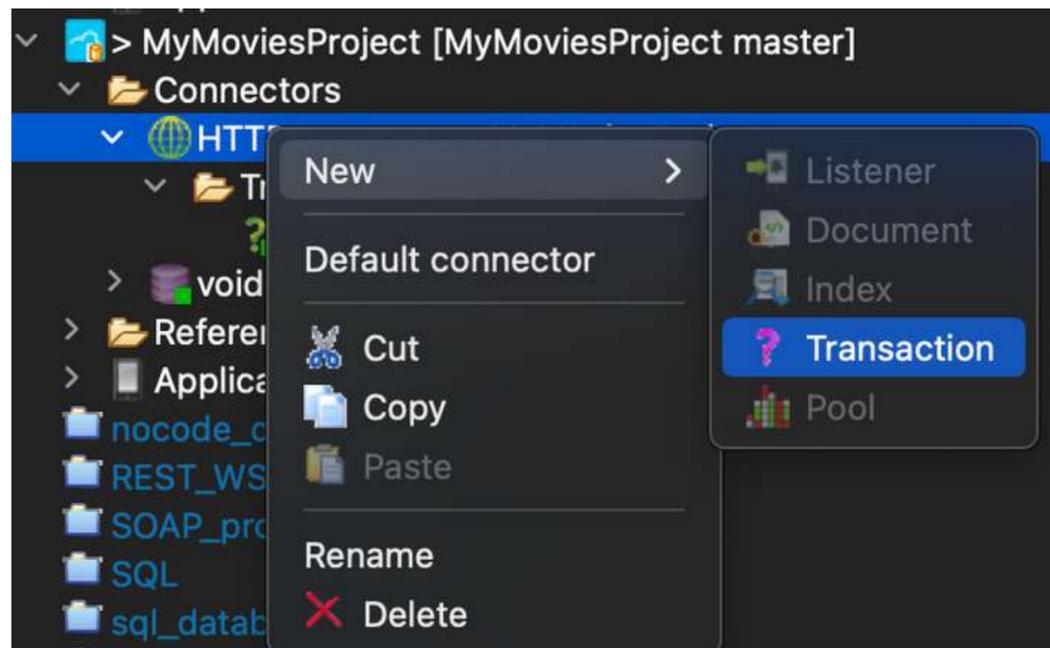


## 3.5 Create a transaction

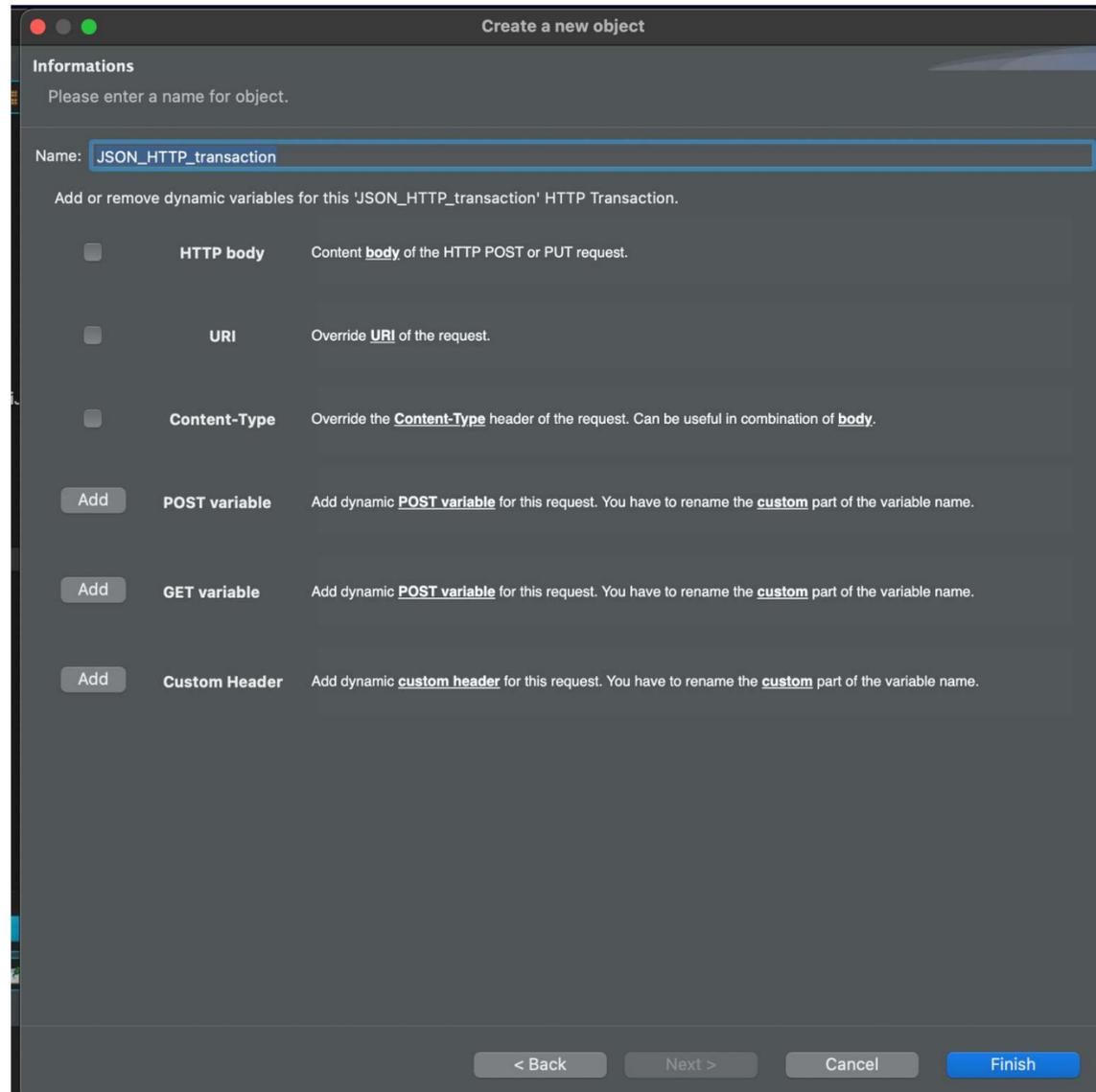
Second Option :

Right-click on the connector,  
then select **New >**, then select **Transaction**.

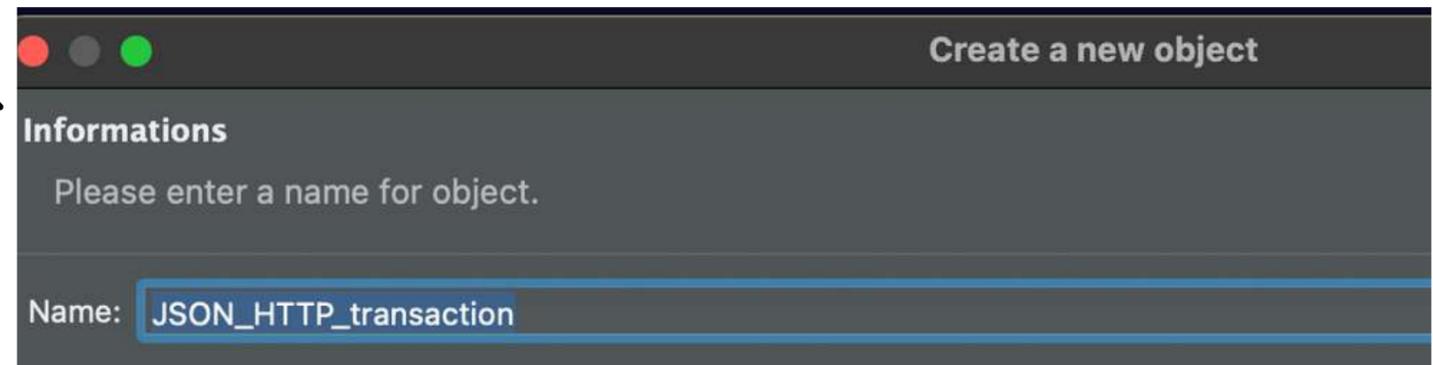
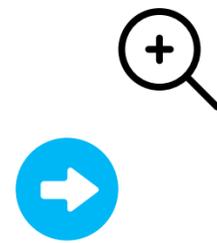
In the **Create a new object window**,  
choose **JSON HTTP transaction**,  
then click on **Next >**.



# 3.5 Create a transaction



Rename the transaction with the name of the request.



Then, follow the same steps as in the first option.



# 3.6 Add a token



To open the web administration console, click on **Convertigo**, then select **Open web administration console**.

Convertigo Window Help

- Engine Preferences
- Studio Preferences
- Configure Registration Account
- Check remote dependencies
- Open Convertigo workspace folder
- Open Convertigo Documentation
- Open Swagger console
- Open web administration console**
- Open the NoCode Databases view
- Open the Tutorial view
- Open the Startup page
- About Convertigo plug-in

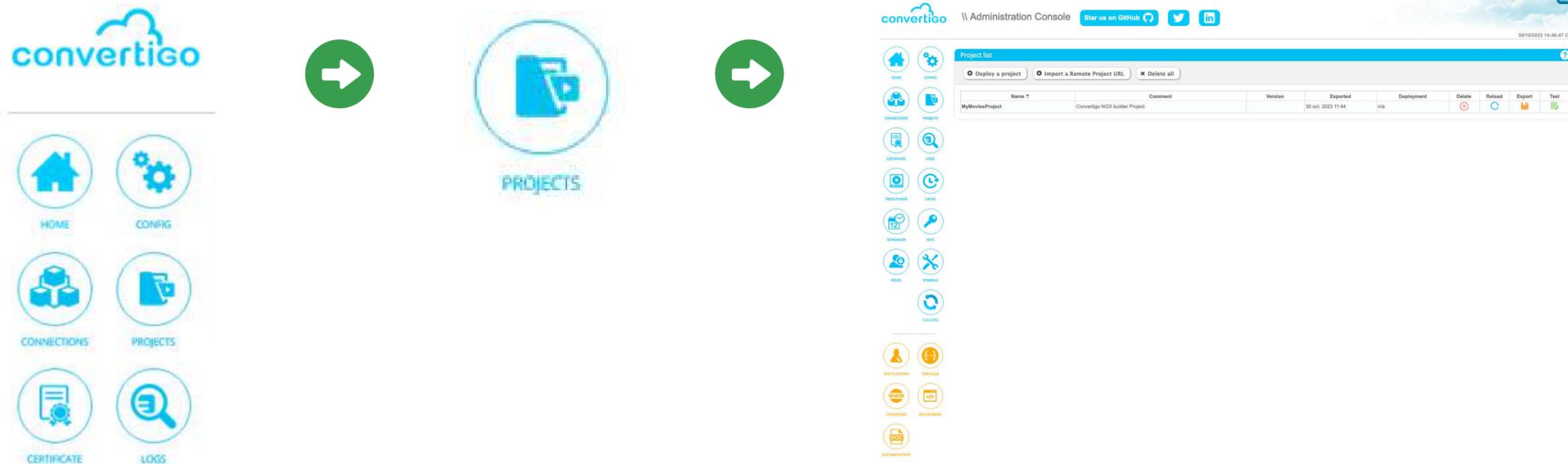


The screenshot shows the 'Administration Console' interface. It features a sidebar with navigation icons for Home, Config, Connections, Projects, Certificate, Logs, Trace Player, Cache, Scheduler, Keys, Roles, Symbols, Full Sync, Test Platform, Swagger, Convertigo, Dev Network, and Documentation. The main content area is divided into three sections: 'Status', 'System Information', and 'Monitor'. The 'Status' section displays system metrics like last startup time and uptime. The 'System Information' section shows details about the host (kitsunezuka), including CPU, OS, network, and memory usage. The 'Monitor' section contains three real-time graphs: 'Memory (MB)' showing maximum, total, and used memory; 'Threads' showing the number of threads; and 'Request duration' showing the average request duration in milliseconds.



# 3.6 Add a token

In the web administration console, click on the icon **PROJECTS** to view the **projects currently opened** in the studio workspace.



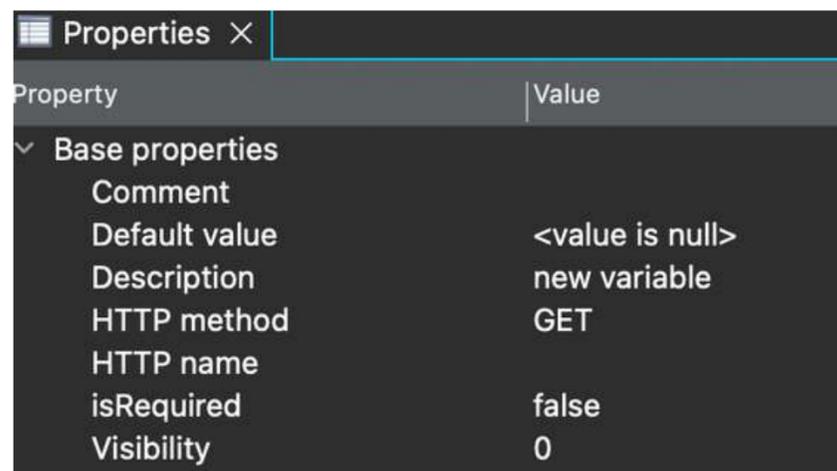
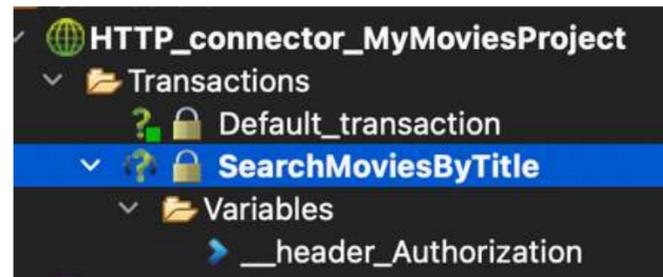
Project list

Name ↑	Comment	Version	Exported	Deployment	Delete	Reload	Export	Test
MyMoviesProject	Convertigo NGX builder Project		30 oct. 2023 11:44	n/a				

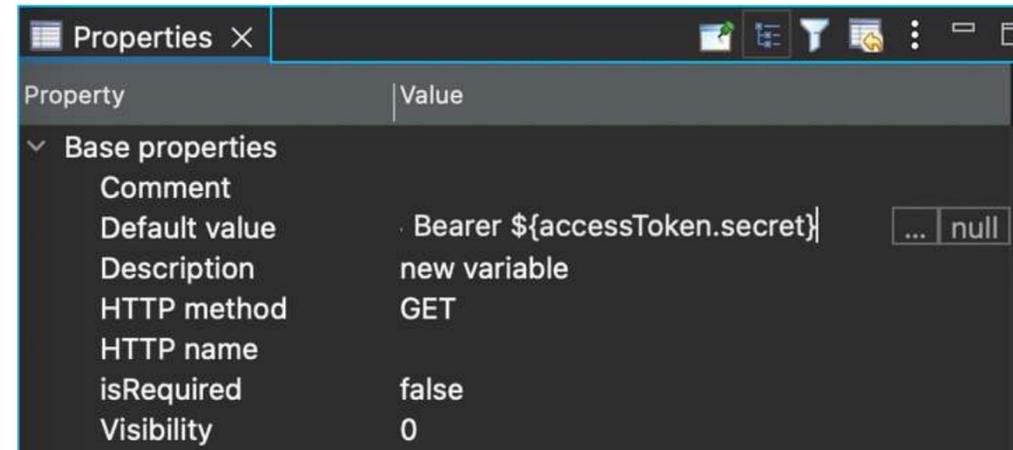


## 3.6 Add a token

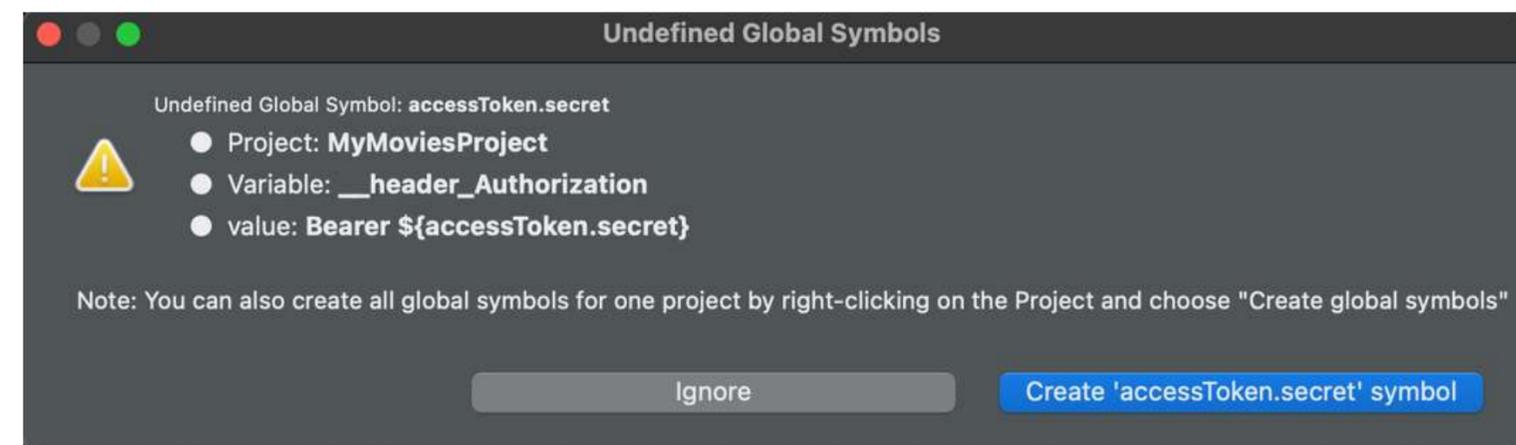
Let's have a look on the properties of the variable `__header_Authorization`.



As **Default value**, enter `Bearer ${accessToken.secret}`.



Validating the value will open the **Undefined Global Symbols** window. Click on **Create 'accessToken.secret' symbol**.



# 3.6 Add a token

In the web administration console, click on **Symbols** to access the Global symbols.



Global symbols

+ Add Symbol   Add Secret Symbol   Import Symbols   Export Symbols   Delete all

Global Symbols values can be fixed string, another Global Symbols or Environment Variables.  
If a symbol is defined for the Default value or if it contains a closing curly braces it must be escaped with a backslash: \}.  
Here is a valid value: `value_${sym1=def_${sym2}}`.

Name	Value	Edit	Delete
accessToken.secret	*****		



accessToken.secret	*****		
--------------------	-------	---	---



Edit



Click on **Edit** to open the **Edit symbol window**.



# 3.6 Add a token

In the **Edit symbol window**, change the **accessToken.secret** value.

Edit secret symbol
✕

In secret mode, the value is stored ciphred and the key automatically ends with **.secret**

Name :

**.secret**

Value :



Edit secret symbol
✕

In secret mode, the value is stored ciphred and the key automatically ends with **.secret**

Name :

**.secret**

Value :



In the **Information window**, a message confirms the changes in the **accessToken.secret** value.

Information
✕

Global symbol 'accessToken.secret' have been successfully edited!



Global symbols
?

+ Add Symbol
🔑 Add Secret Symbol
↓ Import Symbols
↑ Export Symbols
🗑 Delete all

Global Symbols values can be fixed string, another Global Symbols or Environment Variables.  
If a symbol is defined for the Default value or if it contains a closing curly braces it must be escaped with a backslash: \}.  
Here is a valid value: `value_${sym1=def_${sym2}}`.

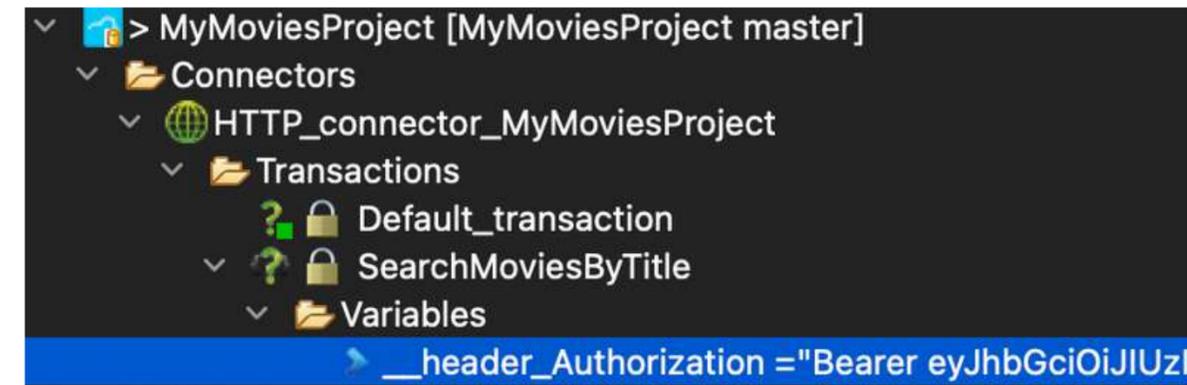
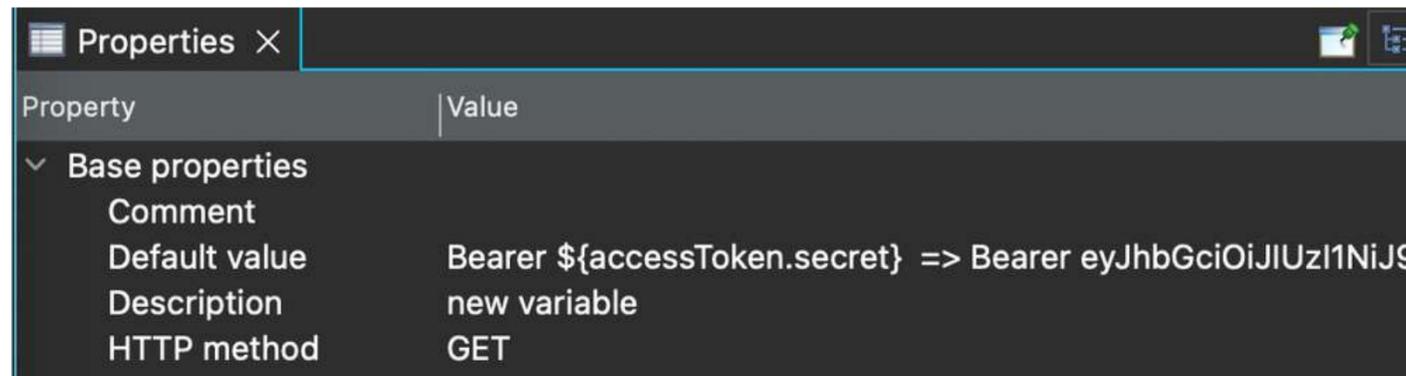
Name	Value	Edit	Delete
accessToken.secret	*****		



## 3.6 Add a token

In the studio, the value of the symbol appears in clear in the Properties of `__header_Authorization`

and in the treeview's variable `_header_Authorization`.



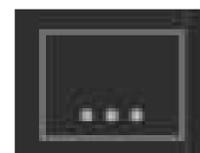
**For security purposes,  
the value of the symbol MUST BE HIDDEN.**

## 3.6 Add a token

To hide the value of the symbol, we need to change the **Visibility** property.

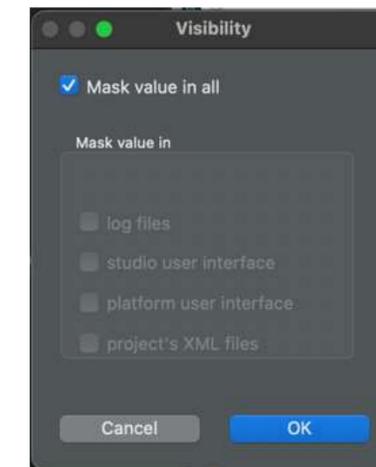
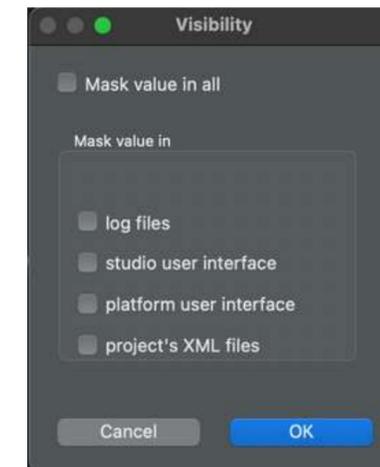
Base properties	
Comment	
Default value	Bearer \${accessToken.secret} => Bearer eyJhbG...
Description	new variable
HTTP method	GET
HTTP name	
isRequired	false
Visibility	0

 **Visibility** 0



Click on the icon at the end of the **Visibility** property line.

The **Visibility** windows appears, click on **Mask value in all**, then click on **OK**.



The value of **\_\_header\_Authorization** is hidden.

In Properties

Base properties	
Comment	
Default value	*****
Description	new variable
HTTP method	GET
HTTP name	
isRequired	false
Visibility	-1

And in the variables folder

```
Variables
  > __header_Authorization = "*****"
```

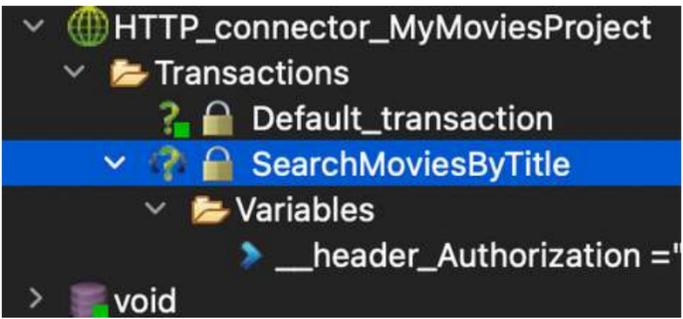
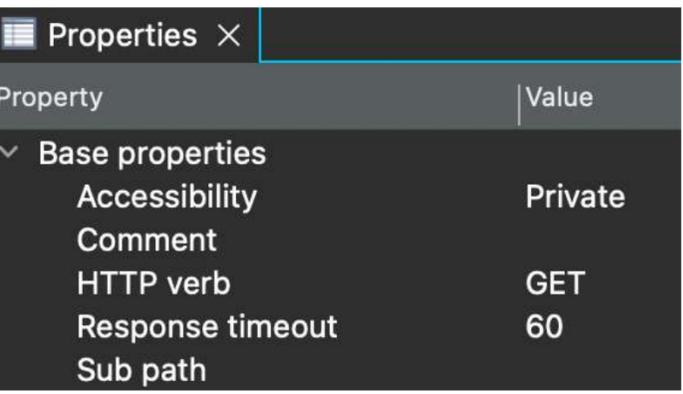


# 3.7 Edit the request path

In the Properties of the transaction,

**edit the Sub path** to include the **request path**:

**search/movie?query={movieTitle}&include\_adult=false&language=en-US&page=1** (as seen in the TMDb API doc)

**Sub path**

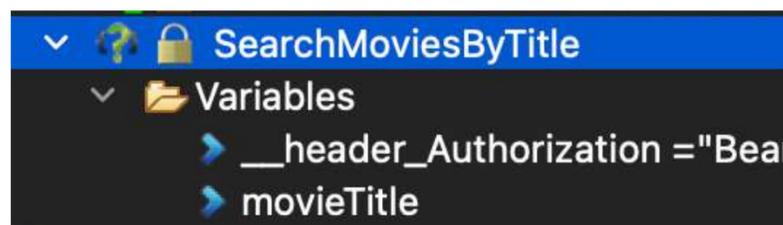


Property	Value
Base properties	
Accessibility	Private
Comment	
HTTP verb	GET
Response timeout	60
Sub path	search/movie?query={movieTitle}&include_adult=false&language=en-US&page=1

To add a **variable part**, enclose it in **curly braces within the path** (in our case, {movieTitle}).



**Sub path** search/movie?query={movieTitle}&include\_adult=false&language=en-US&page=1

This automatically adds the variable to the **Variables folder**.

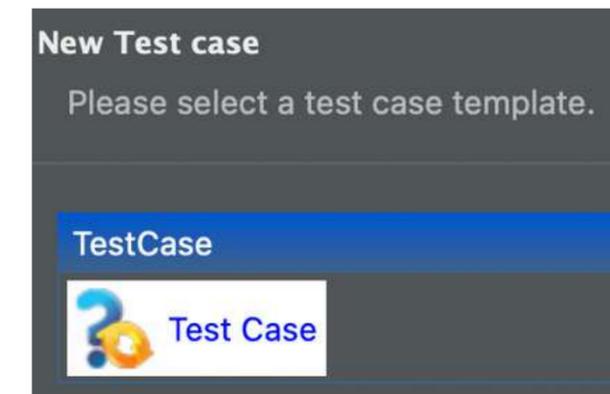
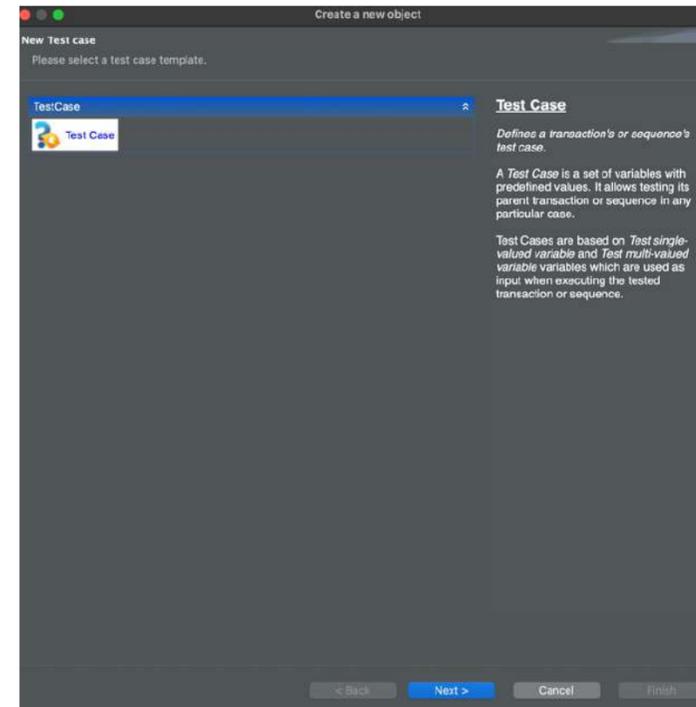
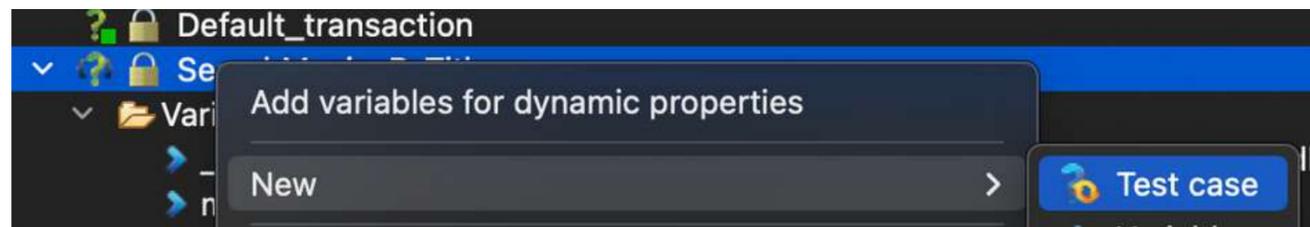


## 3.8 Test the request

To test the request, you need to create a test case.

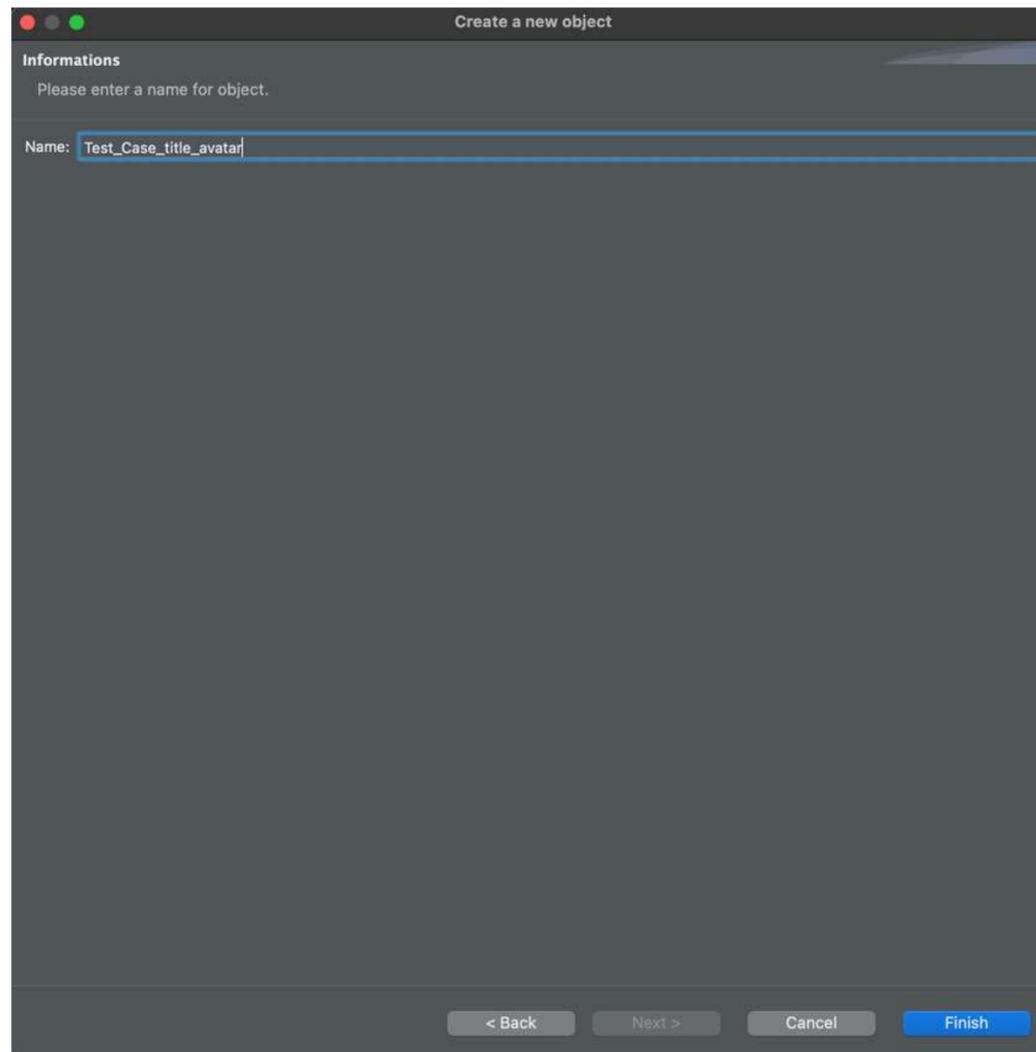
Right-click on the transaction,  
Select **New >**,  
then click on **Test Case**.

In the **Create a new object** window,  
select **Test Case**  
and click **Next**.

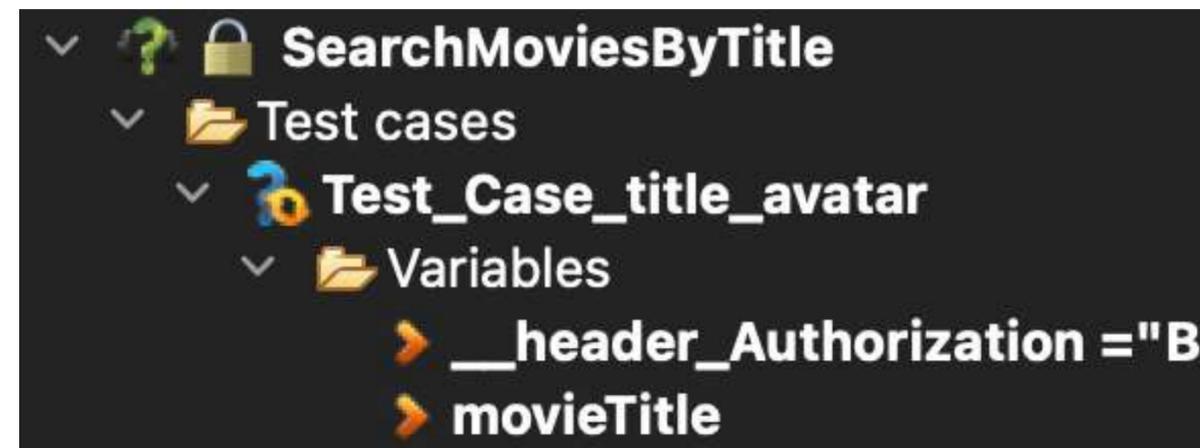


## 3.8 Test the request

Then, enter a name for the test case, and click **Finish**.

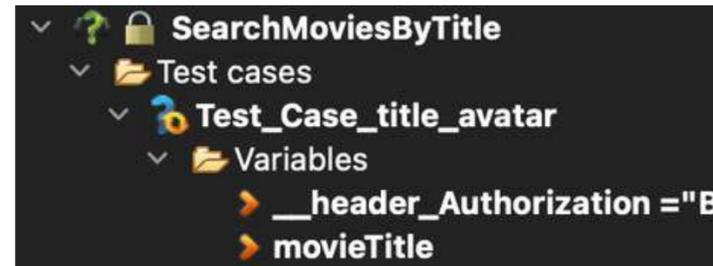


The test case is created in a **Test Cases** folder.

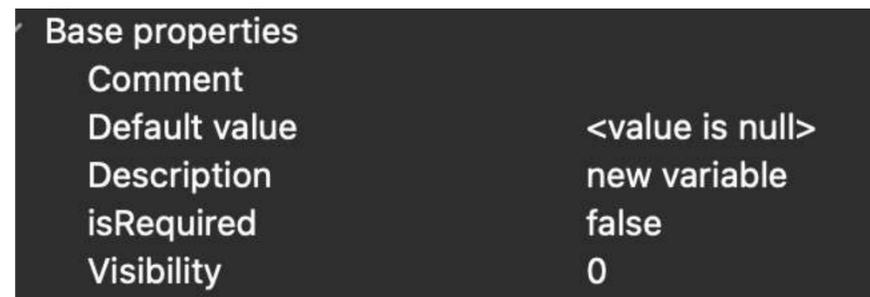


# 3.8 Test the request

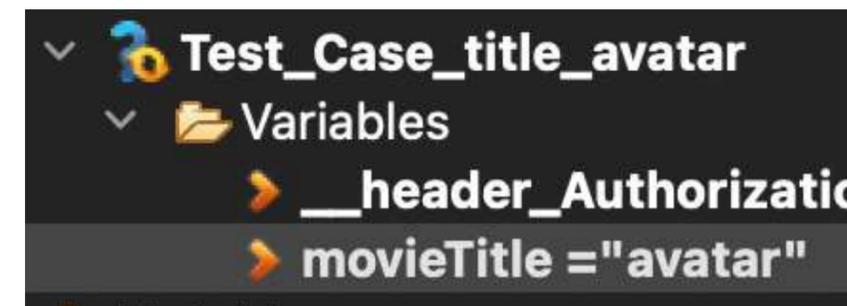
Select the **variable movieTitle** of the **test case**.



In the **Properties**, edit the **Default Value** to enter a search term (in this case, 'avatar').

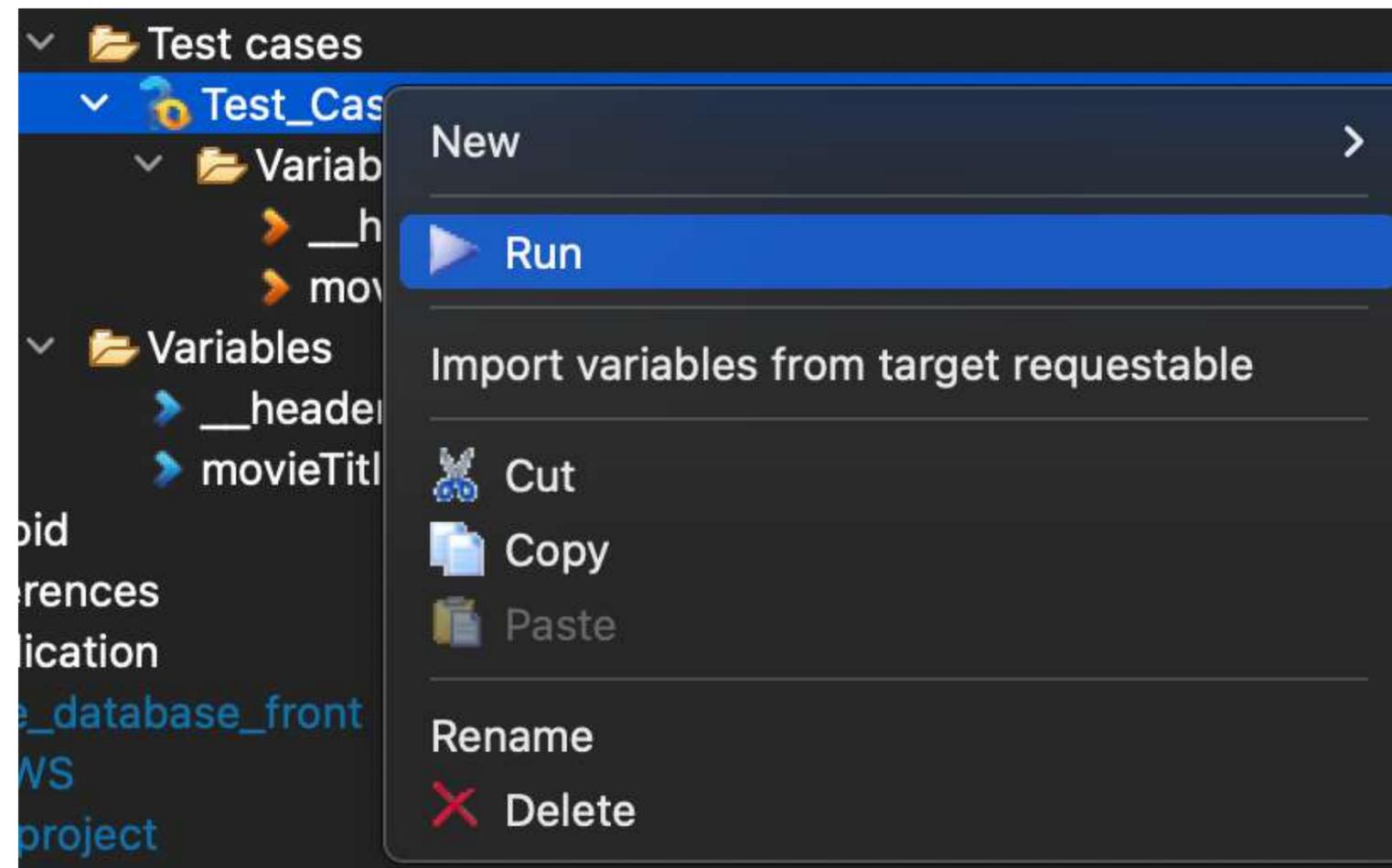


When we **edit the Default value** of the **variable in properties** In the **treeview**, the **value of the variable 'movieTitle'** is **automatically modified**.



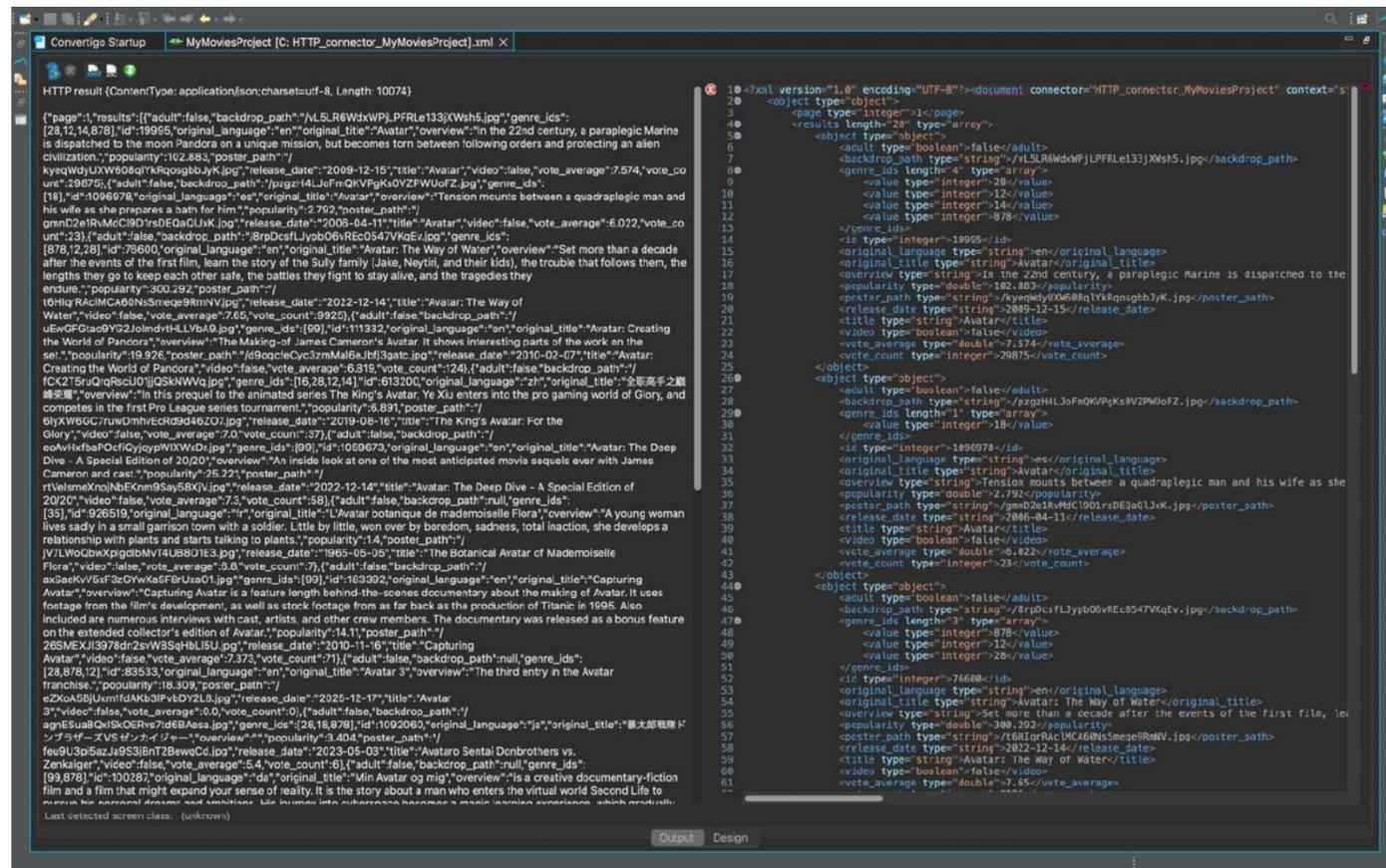
## 3.8 Test the request

To run the test,  
right-click on the test case,  
and click on **Run**.



# 3.8 Test the request

The API response is displayed in XML by default.

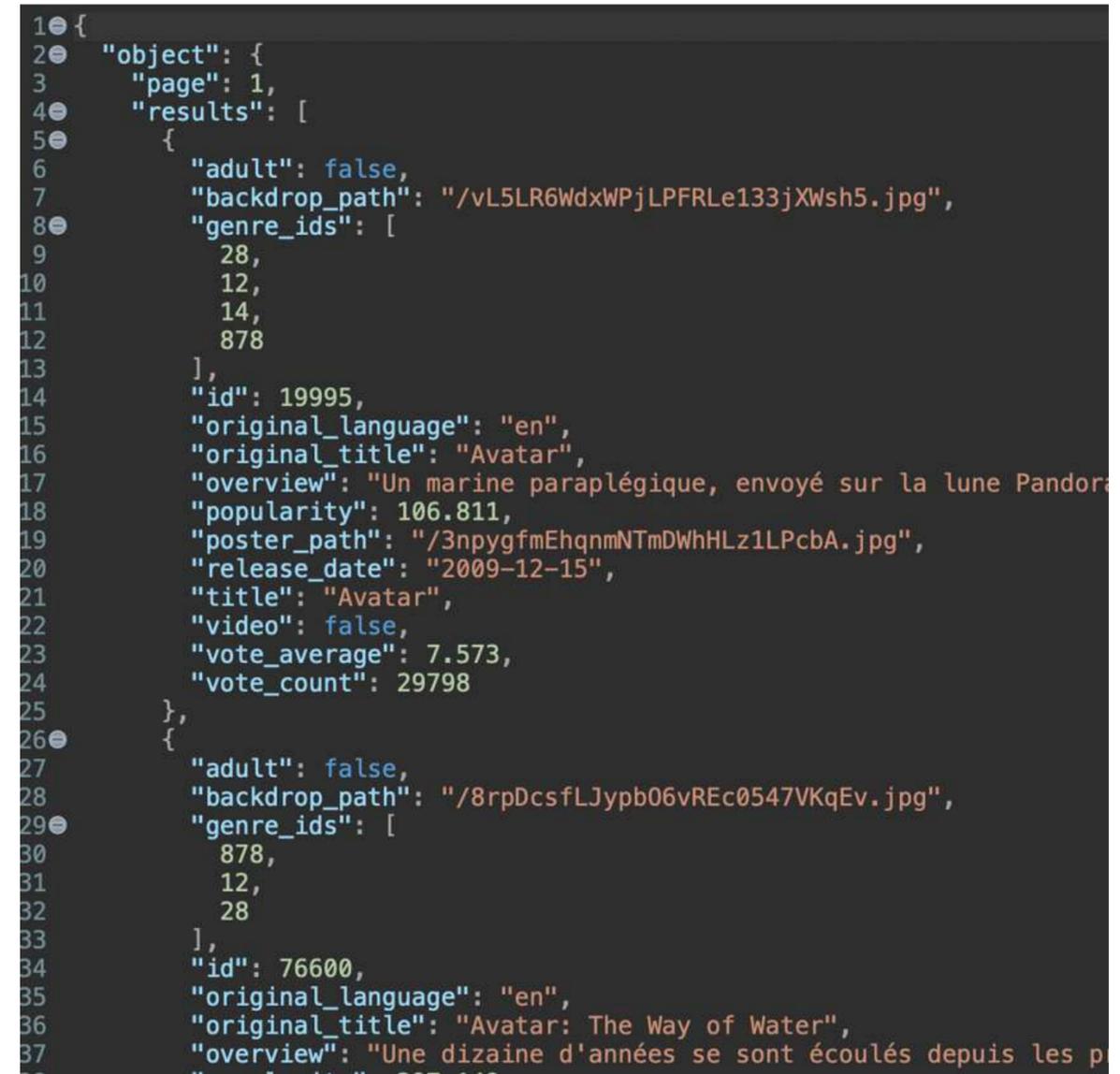
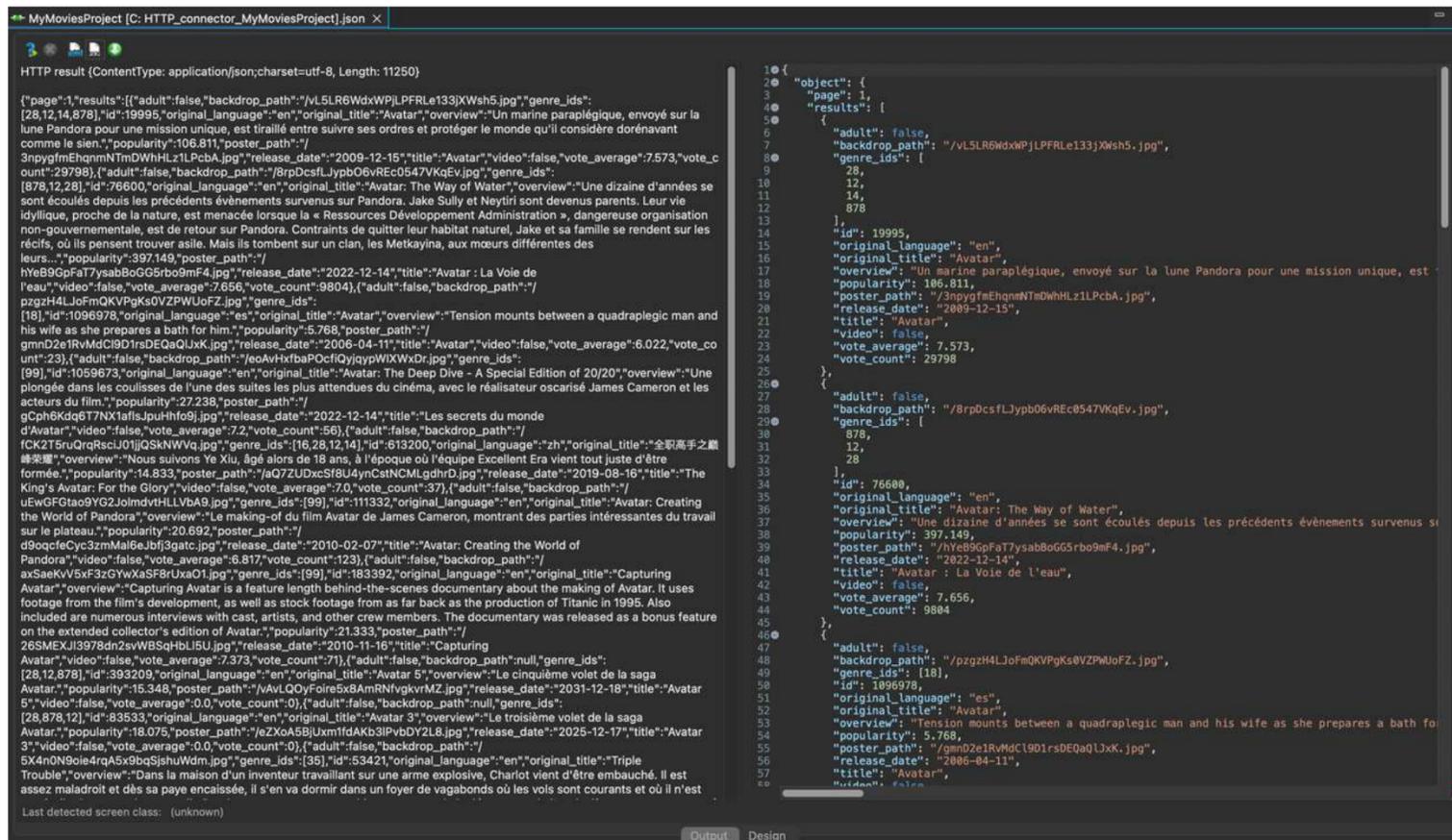
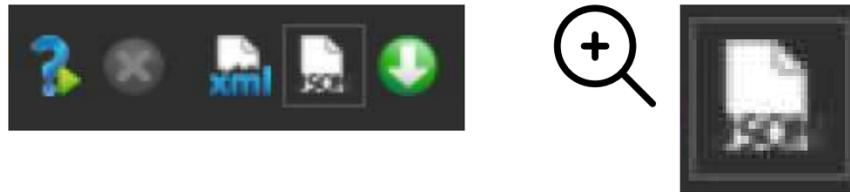


```
1 <?xml version="1.0" encoding="UTF-8"?><document connector="HTTP_connector_MyMoviesT
2 <object type="object">
3 <page type="integer">1</page>
4 <results length="20" type="array">
5 <object type="object">
6 <adult type="boolean">>false</adult>
7 <backdrop_path type="string">/vL5LR6WdxWPjLPFRLe133jXWsh5.jpg</backd
8 <genre_ids length="4" type="array">
9 <value type="integer">28</value>
10 <value type="integer">12</value>
11 <value type="integer">14</value>
12 <value type="integer">878</value>
13 </genre_ids>
14 <id type="integer">1995</id>
15 <original_language type="string">en</original_language>
16 <original_title type="string">Avatar</original_title>
17 <overview type="string">In the 22nd century, a paraplegic Marine is
18 <popularity type="double">124.761</popularity>
19 <poster_path type="string">/kyeqWdyUXW608qLYkRqosgbbJyK.jpg</poster
20 <release_date type="string">2009-12-15</release_date>
21 <title type="string">Avatar</title>
22 <video type="boolean">>false</video>
23 <vote_average type="double">7.574</vote_average>
24 <vote_count type="integer">2992</vote_count>
25 </object>
26 <object type="object">
27 <adult type="boolean">>false</adult>
28 <backdrop_path type="string">/pZgZ4LJoFmQKVPgKs0VZPWUoFZ.jpg</backd
```



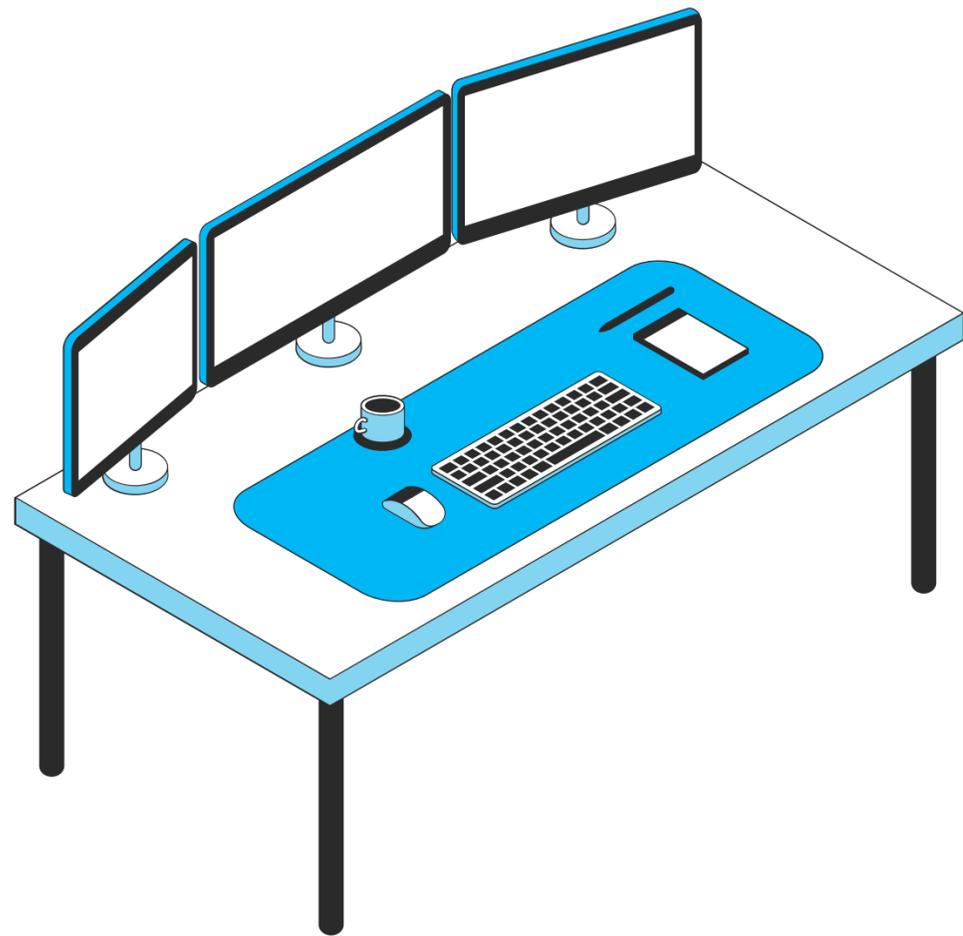
# 3.8 Test the request

Click on the **JSON** button to display the **API response in JSON**.



# 4 – Sequences

How to create a flow of actions.



4.1 Sequences

---

4.2 Steps

---

4.3 XML & XPath

---

4.4 Source Picker

---

4.5 Create a sequence

---

4.6 Call a transaction from a sequence

---

4.7 Create a custom data structure

---

4.8 Test the sequence

# 4.1 Sequences

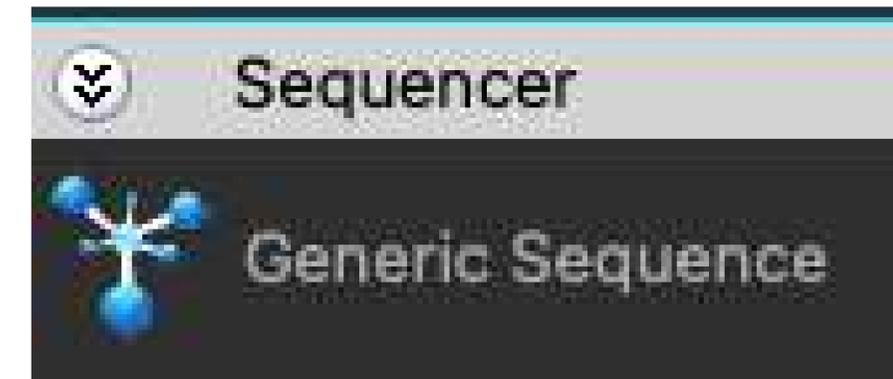
The **Sequence** is a very important **backend object**.  
It is labelled as **Generic Sequence** in the palette.

In Convertigo Low Code Studio, **Sequences** are used to design the **logical flow** and **behavior** of the **backend** of your application by specifying **what actions should occur** and **in what order**.

Sequences allows you to

- create **sequences of actions**
- define **conditions and decision points**
- manage **the order** in which these **actions are executed**
- **define and manage the flow of actions** with a series of successive steps

Object Sequence in the palette



Sequences folder in a project



## 4.2 Steps

Steps are **back-end objects**.

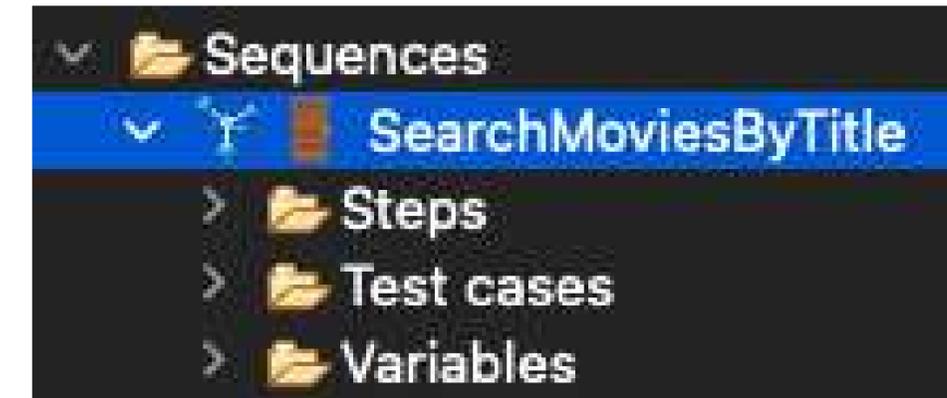
A step is a **fundamental building block** used to **define a specific task, action, or operation** within a sequence.

For example, making an API request, showing a message, performing data manipulation...

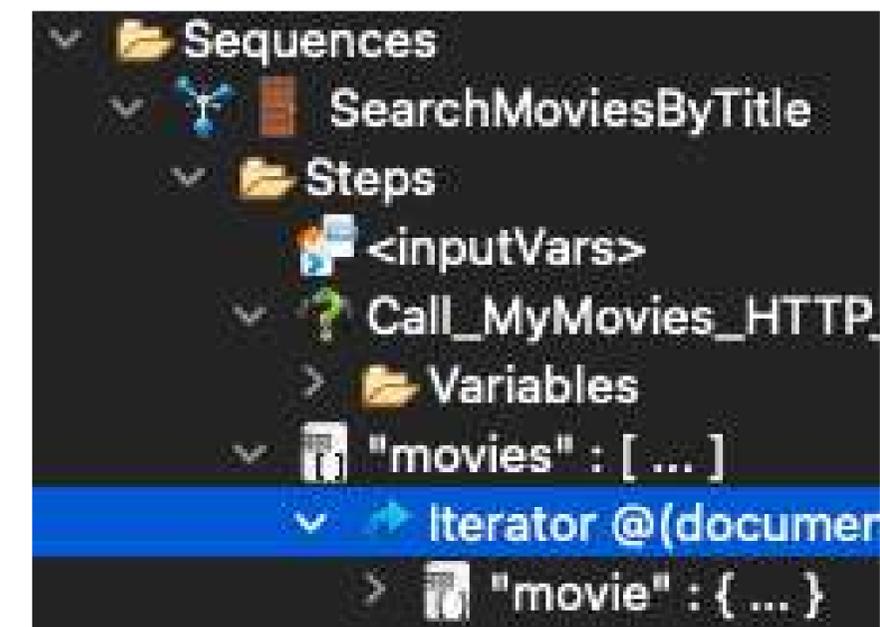
**Steps are organized** to create a **sequence of actions** that the application should perform **in response to certain events or user interactions**.

It allows developers to **define the logic and behavior** of the application in a **structured and modular manner**.

Example of a steps folder in a sequence



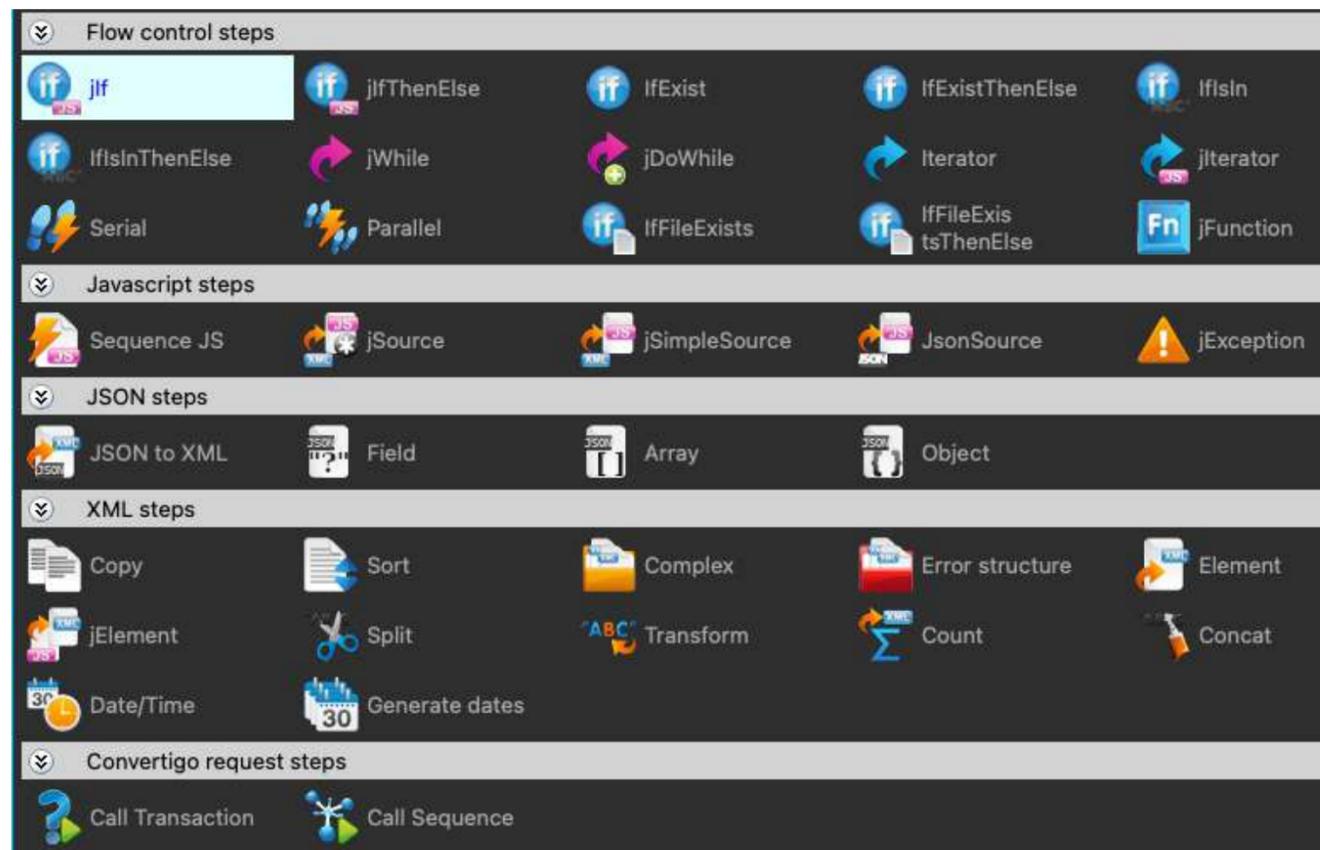
Example of a series of steps in a sequence



# 4.2 Steps

## Categories of Steps

Examples of Steps in the palette



There are different categories of steps :

- **Convertigo request steps** => to call a sequence or transaction
- **Flow Control Steps** => to control the sequence of actions and logic within a sequence
- **Javascript steps** => to incorporate custom JavaScript code in sequences
- **XML steps** => to work with XML data in sequences
- **JSON Steps** => to work with JSON data in sequences
- **HTTP session management** => to manage user sessions in web applications
- **File management steps** => to handle and manipulate files on the local system or server
- Others



# 4.2 Steps

## JSON Steps

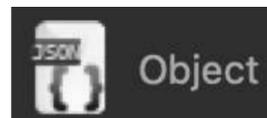
Convertigo provides **JSON steps** to **manipulate and interact with JSON data** in sequences.



### Array – JSON step

When added to a sequence, this step creates an **XML element** (element node) ready to output a JSON Array

- First, you **drag-and-drop the step into a sequence**
- then, you **drag-and-drop the data** you want to manipulate from the Source Picker into the step in the sequence.



### Object – JSON step

When added to a sequence, this step creates a JSON Object.



### Field – JSON Step

When added to a sequence, this step creates a JSON string, number, boolean or null.



## 4.3 XML & XPath

### XML Data Structure

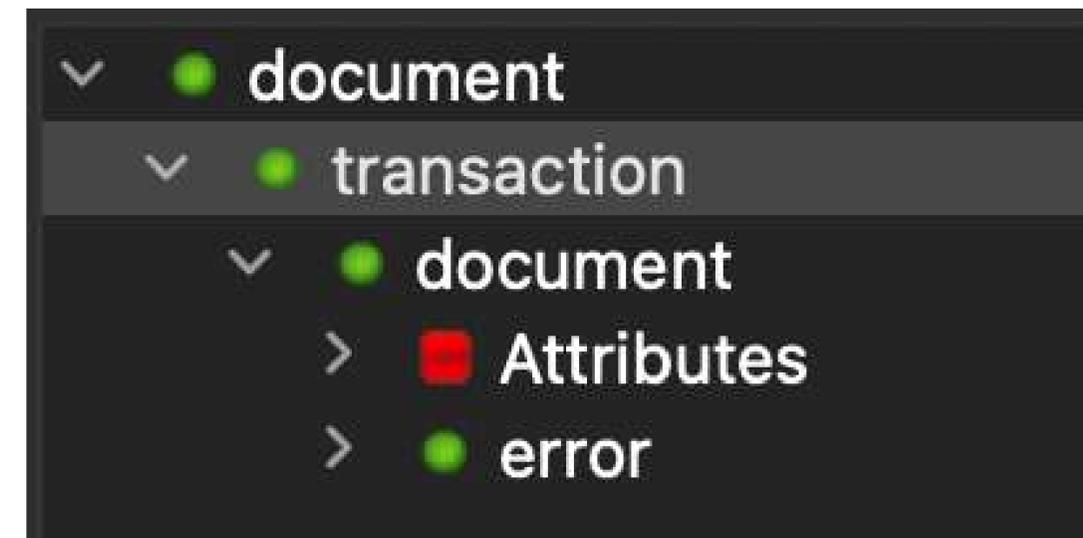
In Convertigo,  
the data structure is **based on XML**  
**regardless of its source.**

The XML data structure follows the standard XML format.

It is organized hierarchically in a **tree structure**  
with one **root element**, the **document**,  
that is the **parent of all other elements.**

Each element has **attributes** and **text content**

Example of XML Data structure in Convertigo



## 4.3 XML & XPath

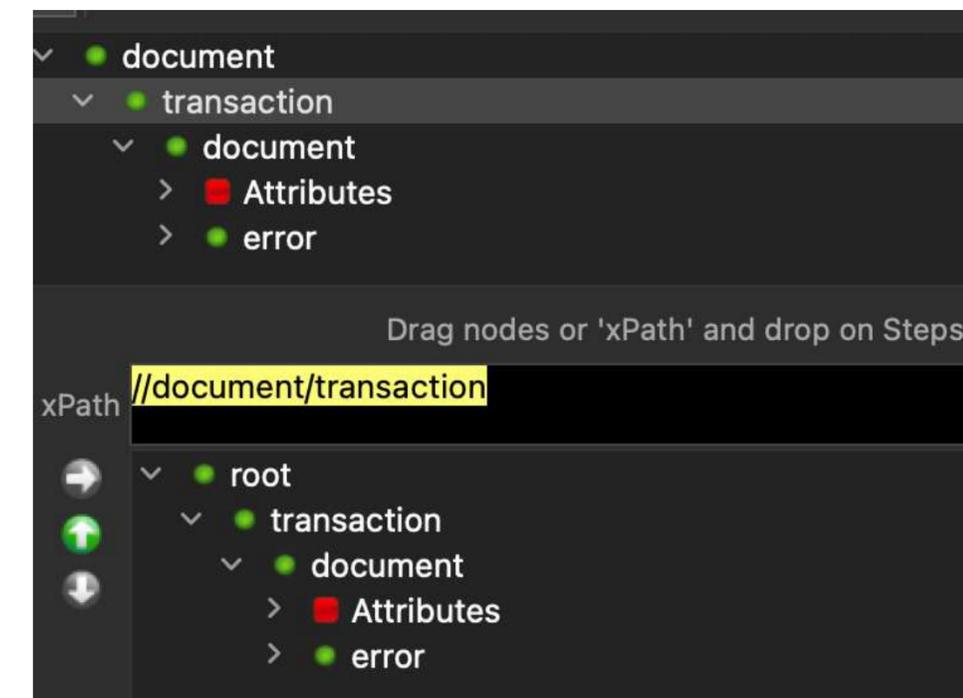
### XPath

XPath is a **language** used for **navigating and querying XML documents**

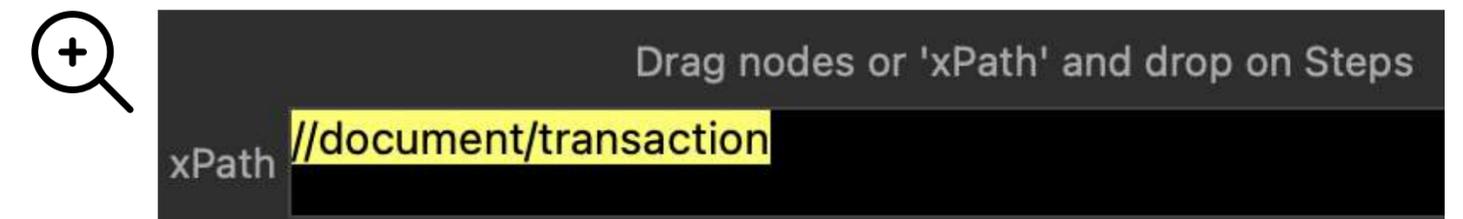
XPath provides a way to pinpoint **specific elements and data** within an **XML structure** by using **path expressions that define the location of nodes.**

**XPath expressions** are used to **identify and traverse these nodes** within an **XML document**, allowing for **data extraction and manipulation.**

Example of XML Data structure & XPath in Convertigo



The screenshot shows a tree view of an XML structure on the left and a configuration panel on the right. The tree view shows a 'document' node containing a 'transaction' node, which in turn contains another 'document' node with 'Attributes' and 'error' children. The configuration panel has a text input field for 'XPath' containing the expression '//document/transaction', which is highlighted in yellow. Above the input field is the instruction 'Drag nodes or 'XPath' and drop on Steps'. Below the input field are navigation icons: a right arrow, a green up arrow, and a down arrow.



This is a zoomed-in view of the configuration panel from the previous screenshot. It shows a magnifying glass icon on the left, the instruction 'Drag nodes or 'XPath' and drop on Steps' at the top, and the 'XPath' input field containing the expression '//document/transaction', which is highlighted in yellow.



## 4.3 XML & XPath

### Nodes

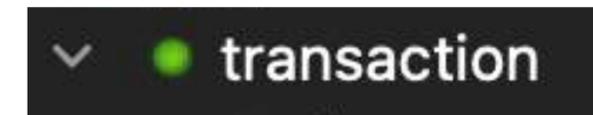
In XML and XPath,

**Nodes** are the **individual components of an XML document**.

There are several types of nodes :

- **element nodes** representing XML elements  
→ marked by a green dot in the XML Data structure in Convertigo
- **attribute nodes** representing attributes of elements  
→ marked by a red square in the XML Data structure
- **text nodes** containing textual content within elements  
→ marked by TxT in the XML Data structure

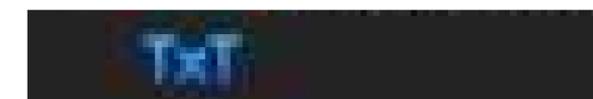
Element node



Attribute node



Text node



## 4.4 Source Picker

### Sources

Each transaction, sequence, and step

- is a **data source** for the next step
- has a property called “output”
- emits data in the source picker

A **source** is defined as a **reference on a step previously existing** in the parent sequence, **associated with an XPath** applied on the step’s result DOM.

At runtime, the **XPath** is applied on the step’s **current execution result XML** and **extracts a list of XML nodes** resulting from this execution.



## 4.4 Source Picker

Each transaction, sequence, and step

- is a **data source** for the next step
- **emits data** in the **source picker**

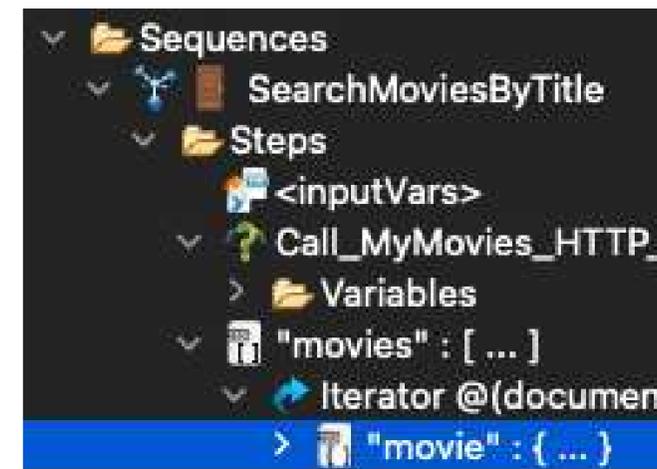
The source picker

- displays the **structure of the data** emitted by a step.
- allows you to **select the XPath** without typing it by **dragging and dropping the node directly into a step**.

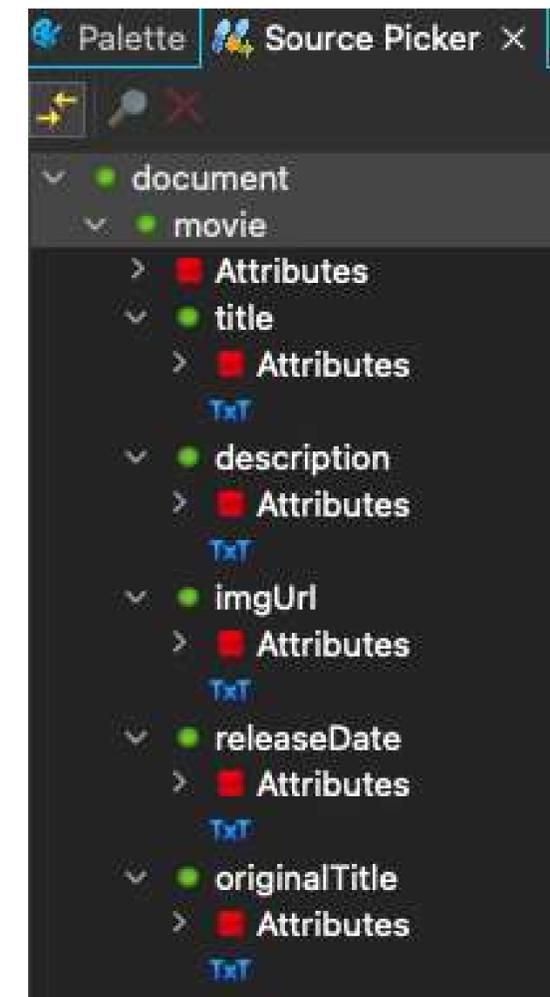
The **XPath** is used as **data path for accessing data**.

Example : step "movie"  
in sequence

```
> <img alt="movie icon" data-bbox="548 318 568 352"/> "movie" : { ... }
```



Data structure of step "movie"  
in source picker



## 4.4 Source Picker

### Output Property

Each transaction, sequence, and step

- has a **property** called “**output**”

The **Output property** defines whether the **XML generated by this step** should be **appended to the resulting XML**.

Properties X	
Property	Value
Expert Output	false

- Set this property to **true** to add the step’s resulting XML to the **sequence’s output XML** (default value for steps generating XML).
- Set this property to **false** to **prevent the steps’s XML result to appear** in the sequence’s output XML. Setting this property to false **does not prevent** the step’s generated XML from being **used as a source by other steps**.



## 4.4 Source Picker

### Output Property

To handle the **data emitted by a step**, there are 2 options :

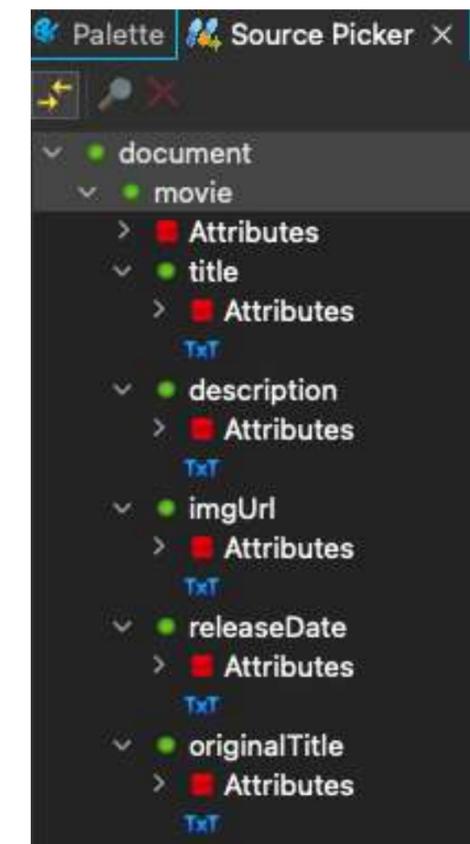
**First option** : If you need the **whole data emitted by a step**

1. Put '**output**' on '**true**'
2. The step emits data in the response

**Second option** : If you need to **filter the data** and keep only specific data

1. Put '**output**' on '**false**'
2. The step doesn't emit in the response but still **emits in the source picker**.
3. You **select the data** you need in the **source picker** by **drag-and-dropping it in a sourceable step**
4. The following step can connect to this source through it.

Properties	
Property	Value
Expert Output	false



# 4.4 Source Picker

## Transaction data structure

Calling a **transaction** brings back **data** with a **structure** described in the **source picker**.

In other steps, the **output structure** is always the same.

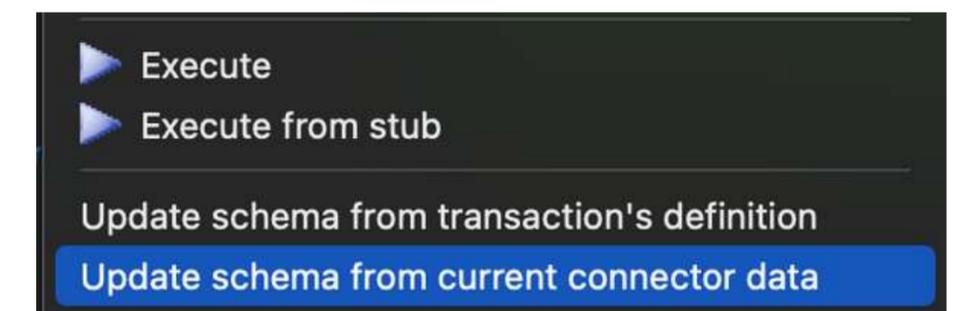
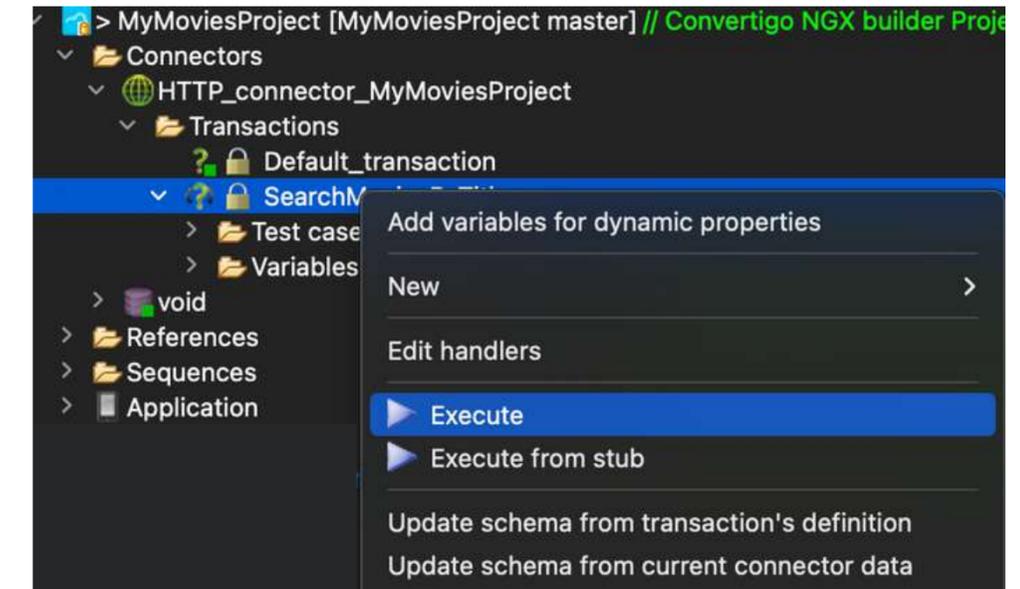
In a **transaction call**, the **output structure** is unknown.

→ To discover it, you need to **execute the transaction once**.

→ Then retrieve the structure with **Import data structure from current connector data..**



As good practice, this should be done when the transaction is created, before creating the sequence.

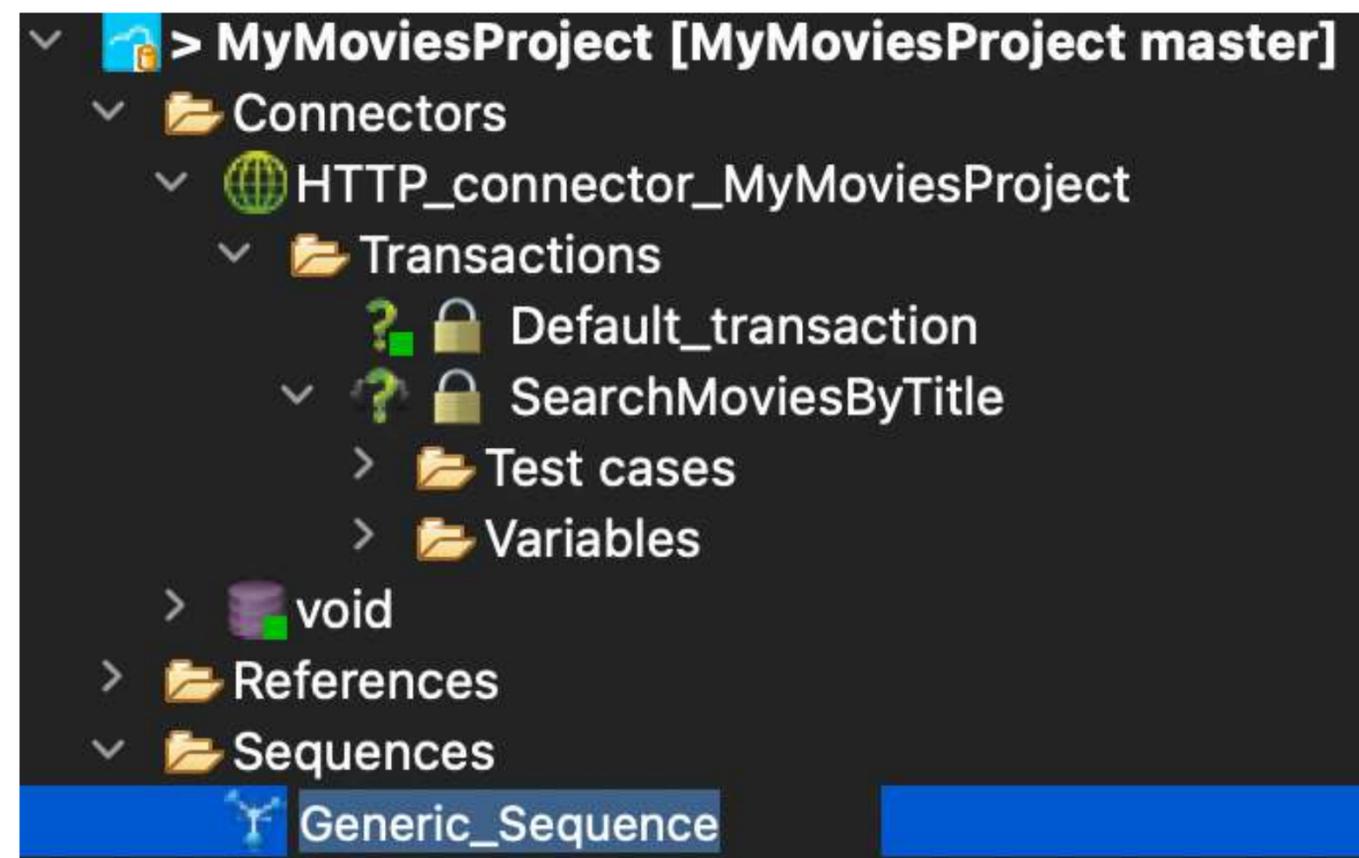
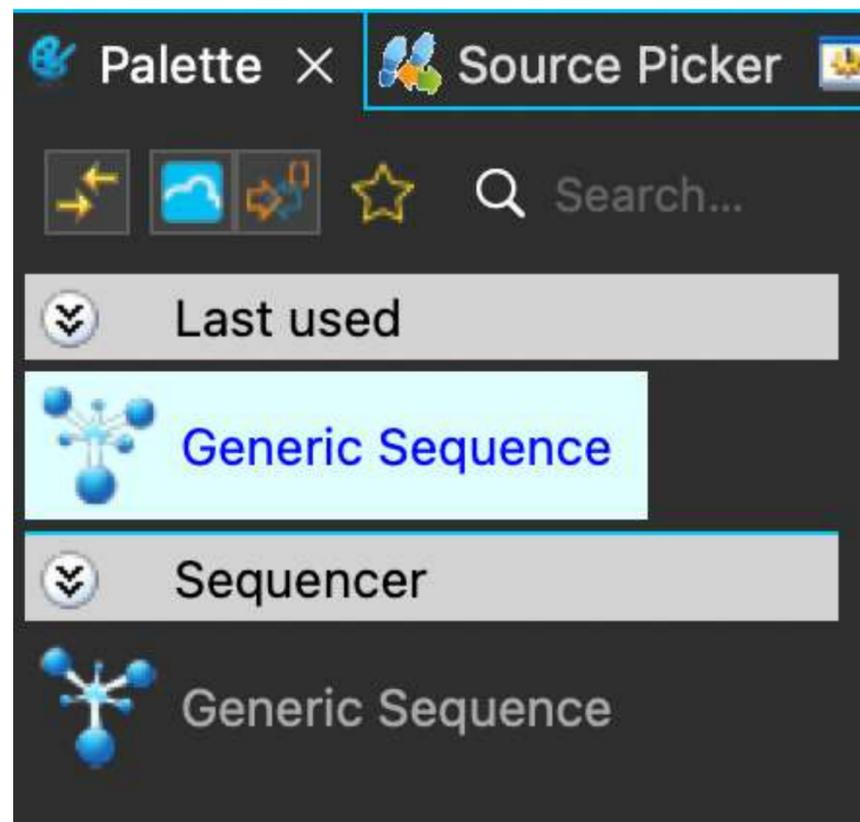


## 4.5 Create a sequence

To create a sequence, you have several options.

First option:

You can **drag and drop a Generic Sequence** from the palette in the tree structure and rename it.

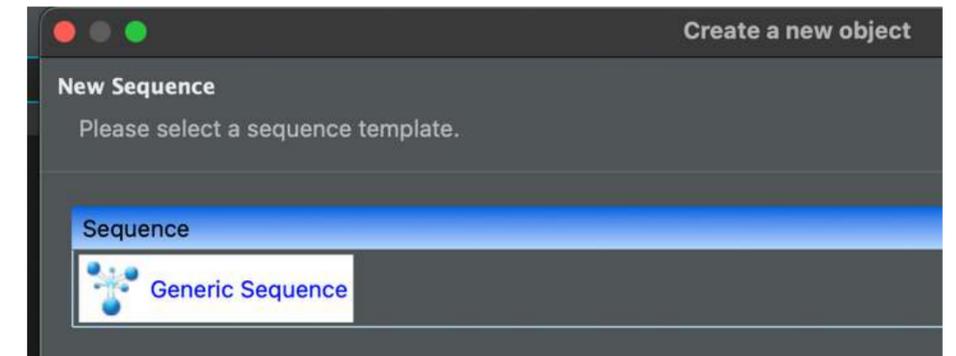
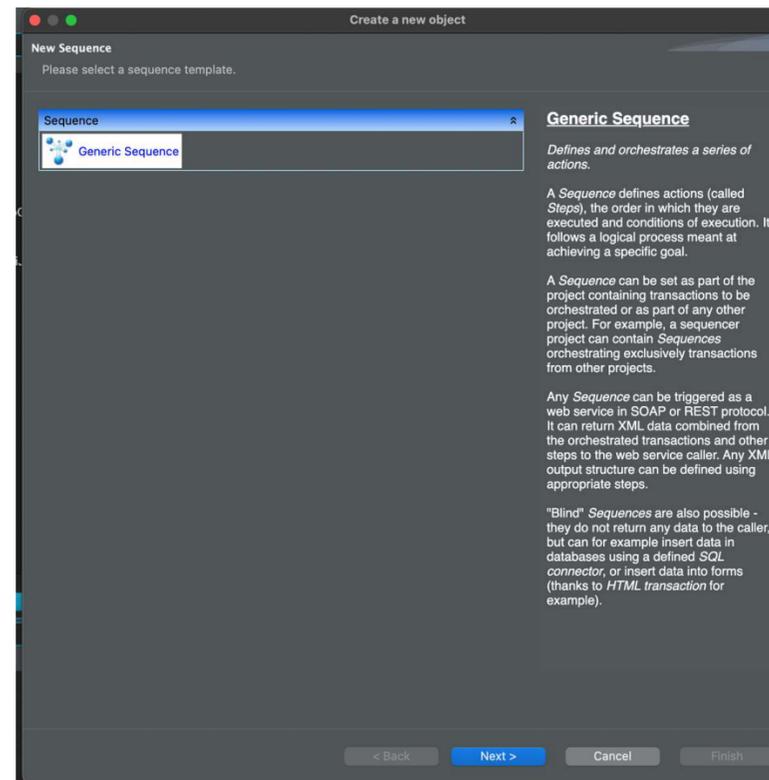
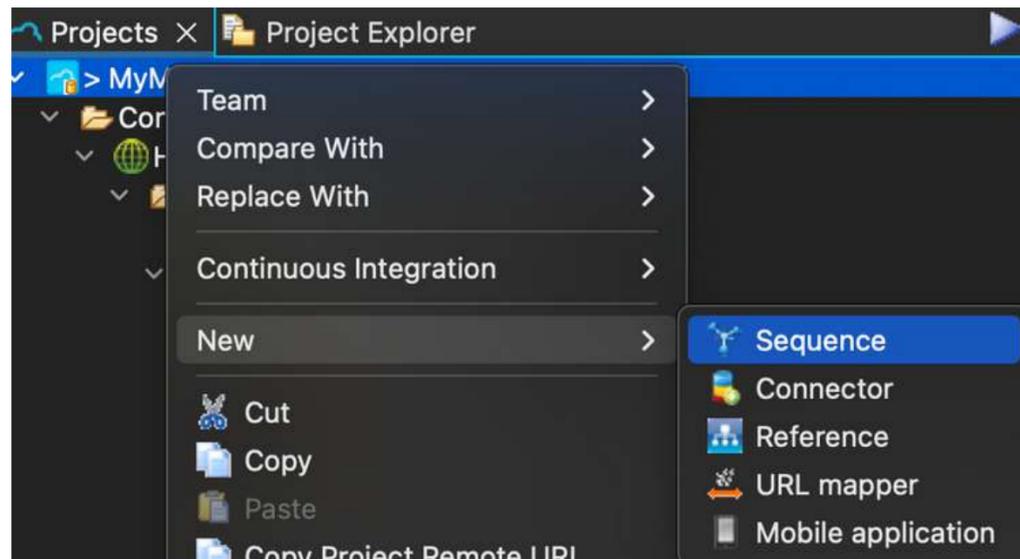


# 4.5 Create a sequence

## Second option:

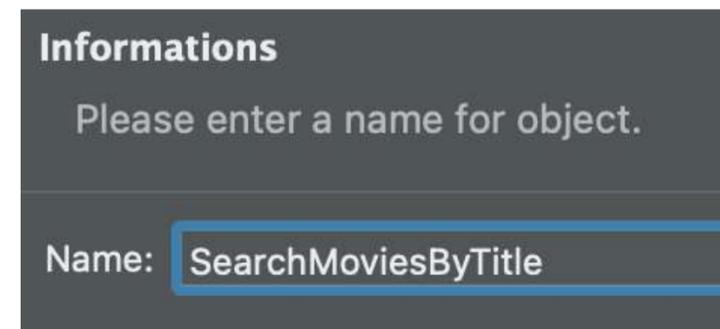
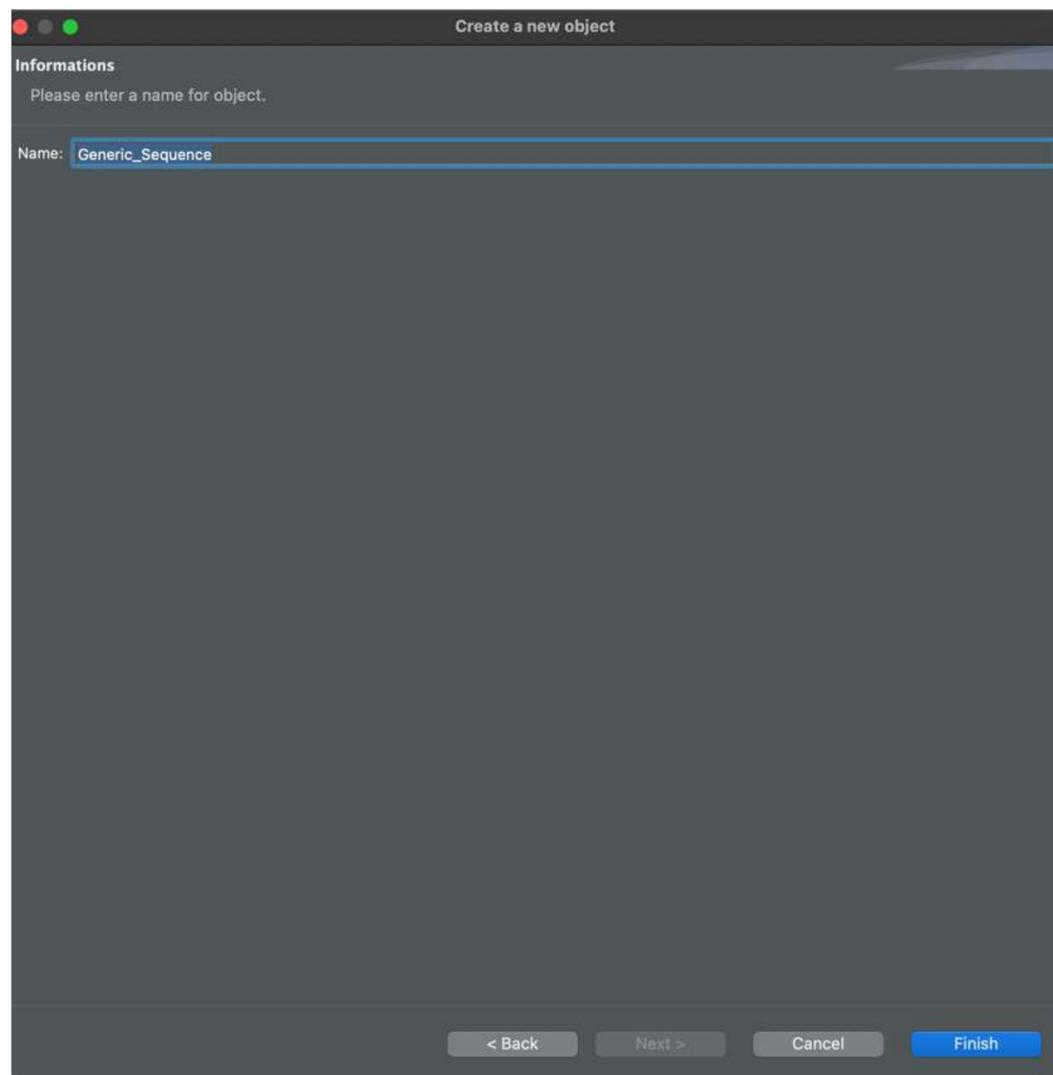
To create a sequence, you can **right-click on the project**, select **New >**, then click on **Sequence**.

The **Create a new object window** appears. In the **Create a new object window**, select **Generic Sequence**, then click on **Next >**

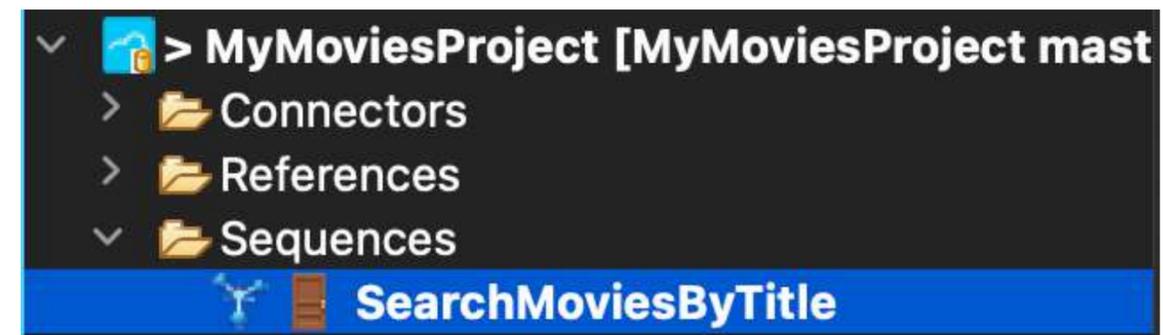


## 4.5 Create a sequence

In the **Create a new object** window, rename the sequence and click on **Finish**.



A **Sequences** folder and the **created sequence** appear in the tree structure.

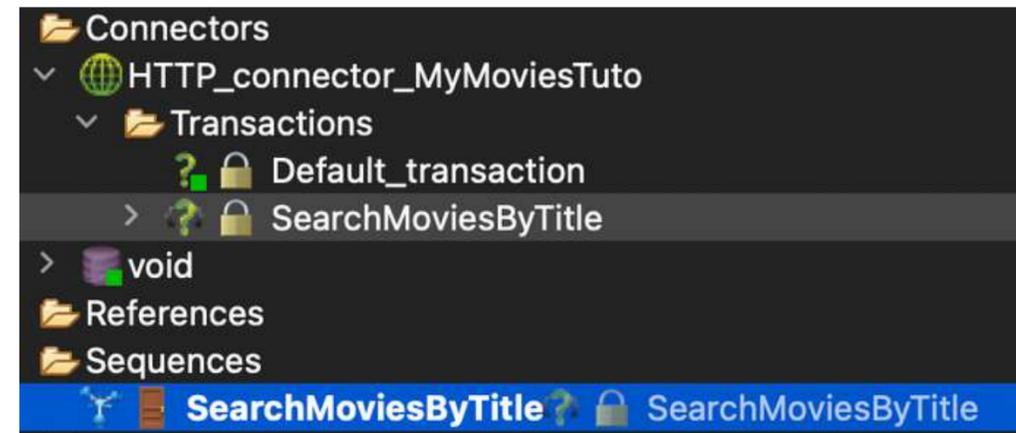
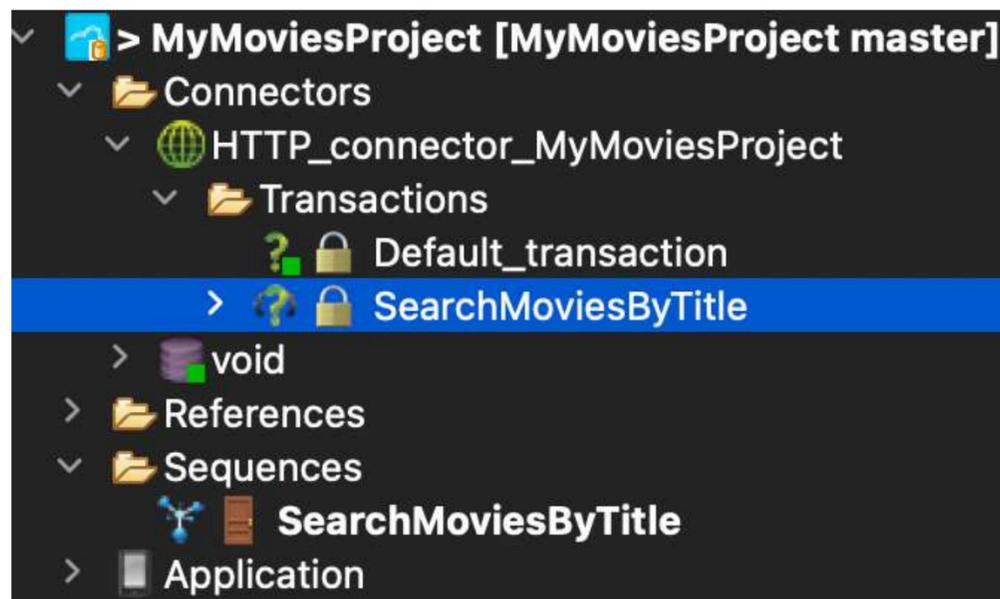


# 4.6 Call a transaction from a sequence

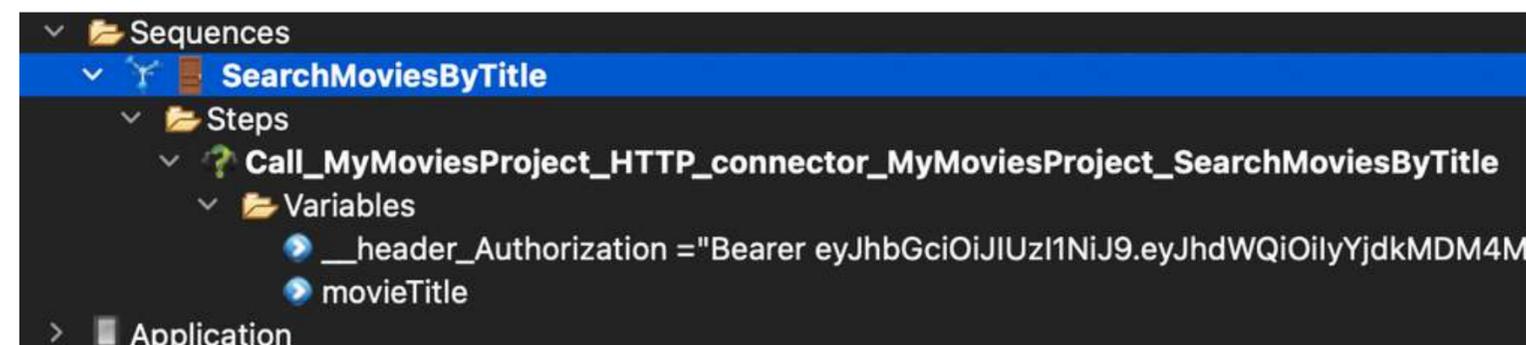
## Import the transaction in the sequence

Once the sequence is created, you need to **import the transaction in the sequence.**

Drag-and-drop the transaction in the sequence while clicking on **Option** in MacOs or **Control** in Windows.



This creates a **Steps folder** where a **call to the transaction** appears.



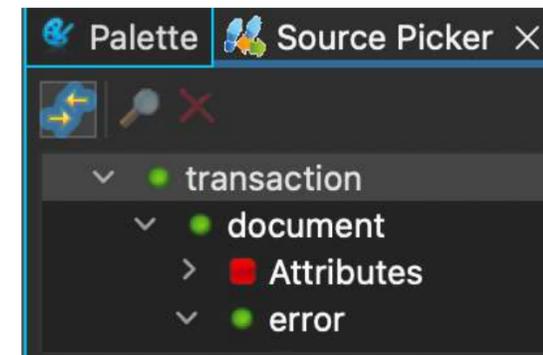
# 4.6 Call a transaction from a sequence

## Update the transaction schema

Double-click on the Call Transaction step to display the source picker.

Reminder : This step can and usually should be done just after creating the transaction.

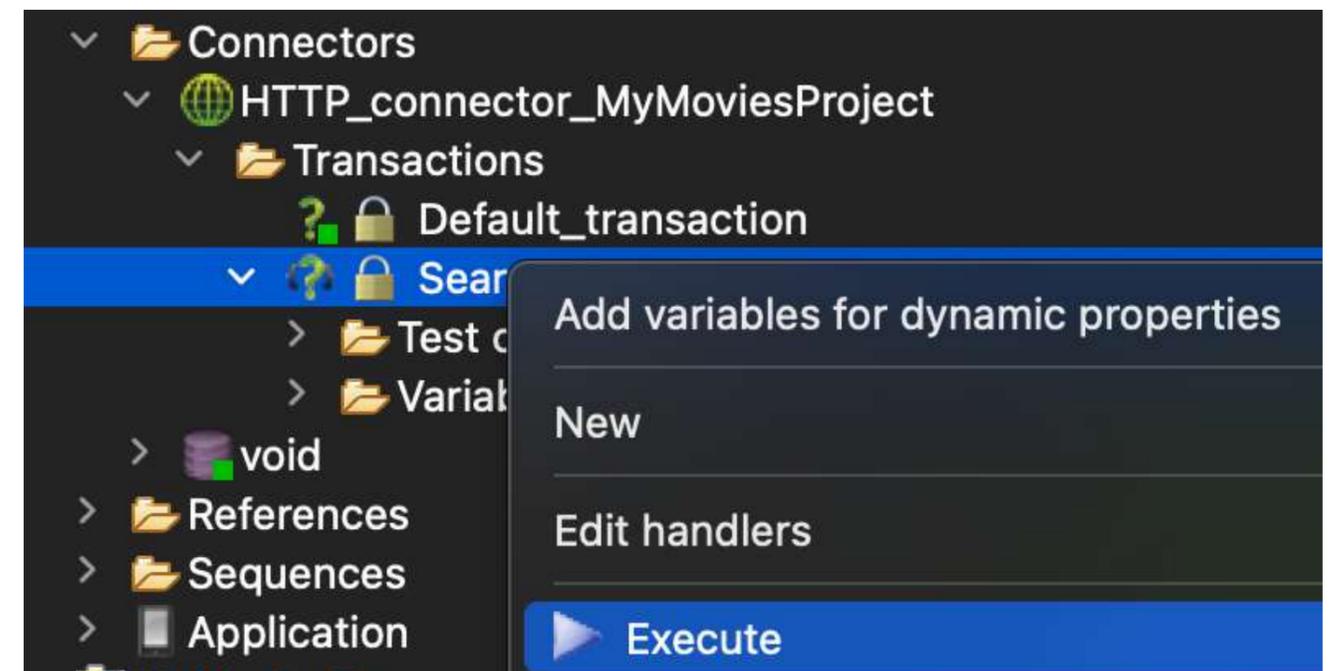
Call\_MyMoviesProject\_HTTP\_connector\_MyMoviesProject\_SearchMoviesByTitle



The **schema shown in the picker does not contain response elements**, you need to **update the transaction schema.**

There are 2 cases : **a transaction with or without variables.**

If the transaction **doesn't need variables**, you can right-click on the source transaction and choose **Execute** to generate response data.

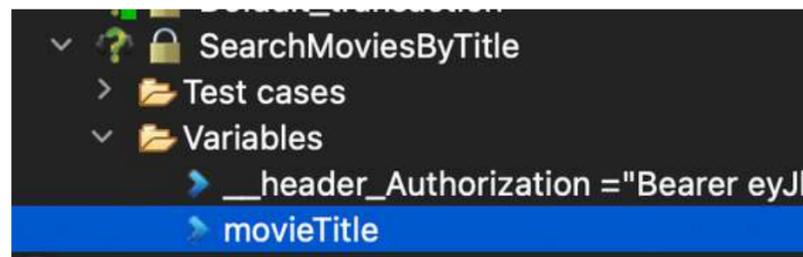


# 4.6 Call a transaction from a sequence

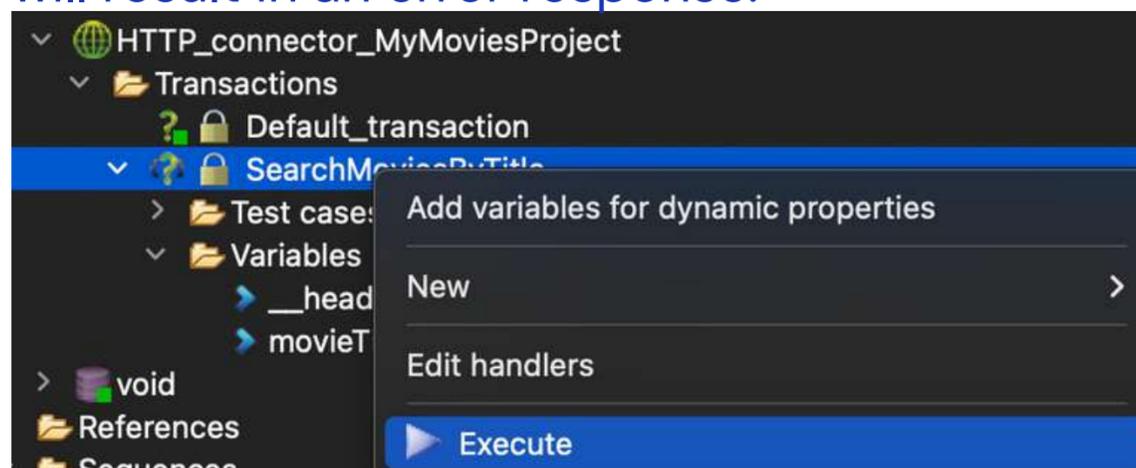
## Update the transaction schema

In our case, we have 2 variables :

- **\_\_header\_Authorization** which has already a value
- **movieTitle** whose value is empty.



Executing the transaction as it is will result in an error response.



```

1 {
2   "error": {
3     "code": "-1",
4     "message": "An unexpected error has occurred while the execution of
5     "details": "Cannot invoke \"String.indexOf(int)\" because \"s\" is
6     "context": "",
7     "exception": "com.twinsoft.convertigo.engine.EngineException",
8     "stacktrace": "com.twinsoft.convertigo.engine.EngineException: An
9     "attr": {
10      "connector": "HTTP_connector_MyMoviesProject",
11      "project": "MyMoviesProject",
12      "transaction": "SearchMoviesByTitle",
13      "type": "c80"
14    }
15  }
16 }

```

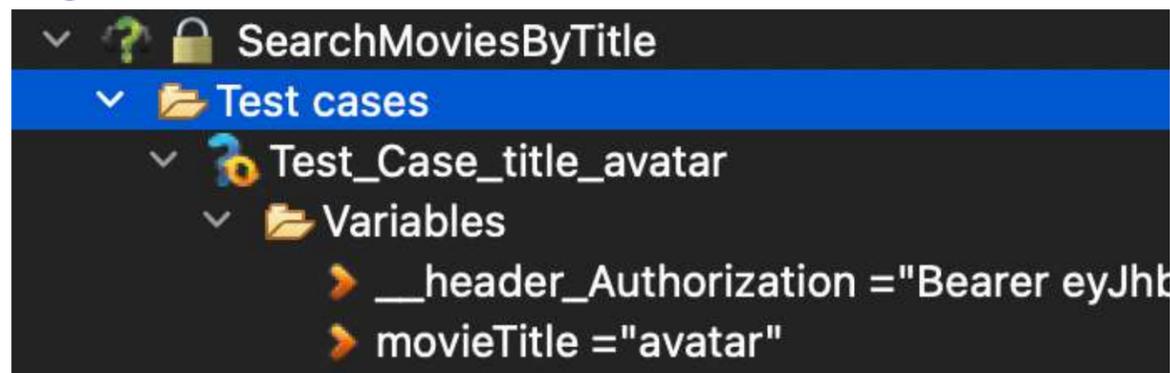


# 4.6 Call a transaction from a sequence

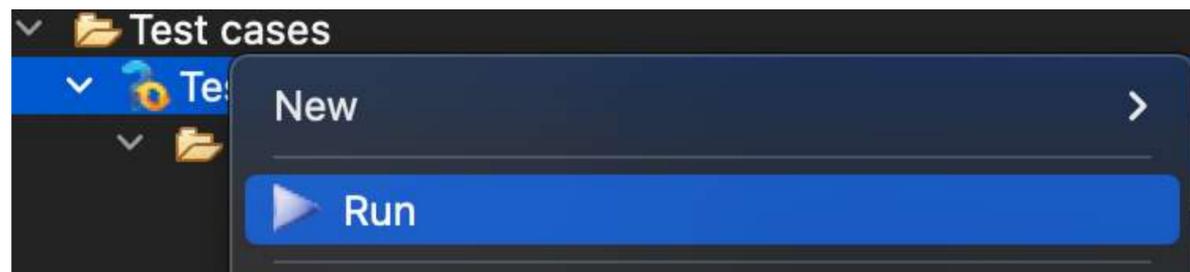
## Update the transaction schema

To get a valid response, you need to use the test case you created before in the transaction SearchMoviesByTitle. In this test case, the variable **movieTitle** has already a value.

Right-click on the test case.



Choose **Run** to generate response data.



The results are displayed in the editors panel.

```

1 {
2   "object": {
3     "page": 1,
4     "results": [
5       {
6         "adult": false,
7         "backdrop_path": "/vL5LR6WdxWPjLPFRLe133jXWsh5.jpg",
8         "genre_ids": [
9           28,
10          12,
11          14,
12          878
13        ],
14        "id": 19995,
15        "original_language": "en",
16        "original_title": "Avatar",
17        "overview": "Un marine paraplégique, envoyé sur la lune F",
18        "popularity": 124.761,
19        "poster_path": "/3npygfmEhqnmNTmDWhHLz1LPcbA.jpg",
20        "release_date": "2009-12-15",
21        "title": "Avatar",
22        "video": false,
23        "vote_average": 7.574,
24        "vote_count": 29927
25      },
26      {
27        "adult": false,
28        "backdrop_path": "/8rpDcsfLJypb06vREc0547VKqEv.jpg",
29        "genre_ids": [
30          878,
31          12,
32          28
33        ],
34        "id": 76600,
35        "original_language": "en",
36        "original_title": "Avatar: The Way of Water",

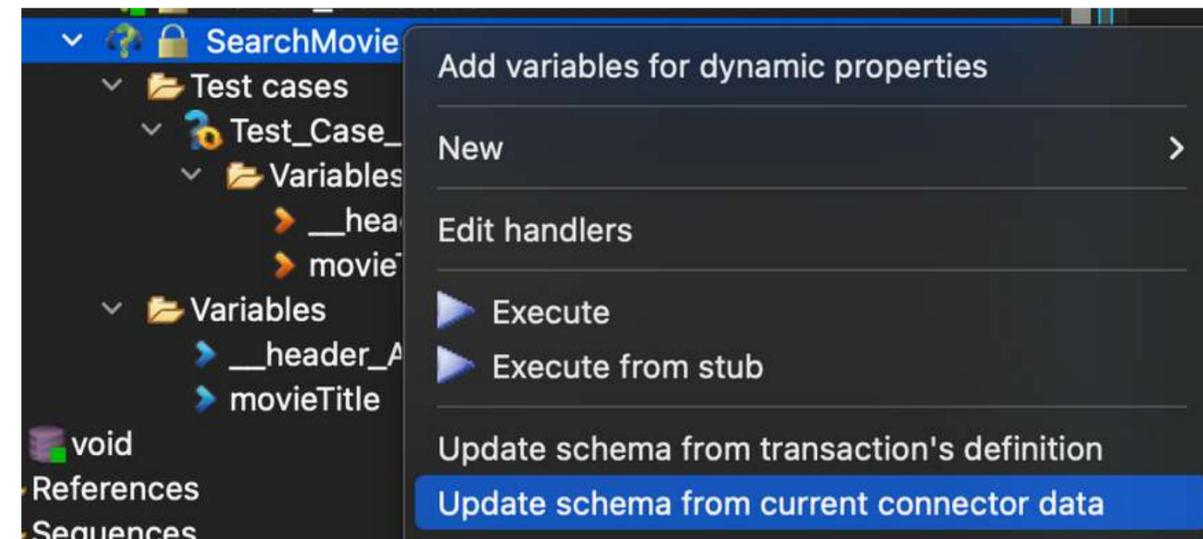
```



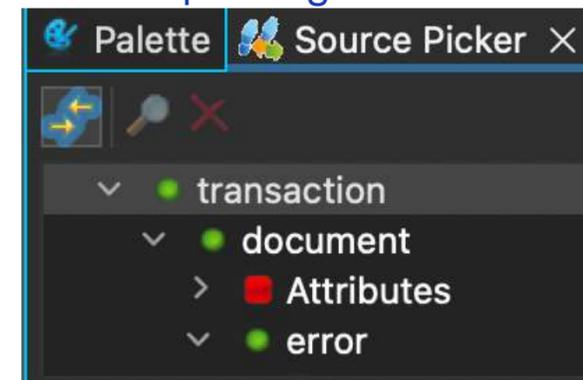
# 4.6 Call a transaction from a sequence

## Update the transaction schema

Right-click on the source transaction **SearchMoviesByTitle** and select **Update schema from current connector data**.



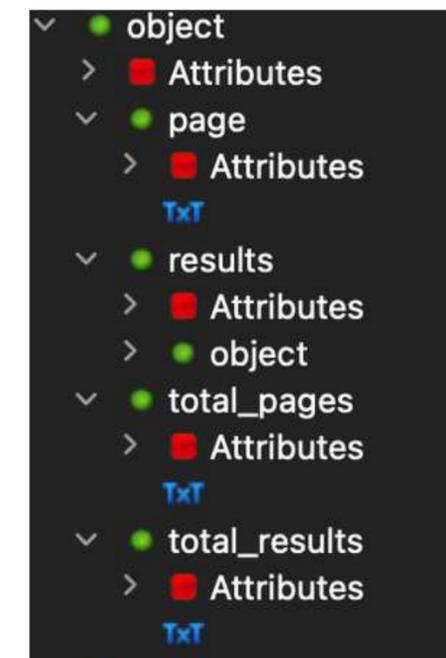
Before updating the schema



Double-click once again on the **Call Transaction Step** to display the **updated transaction schema** in the Source Picker.



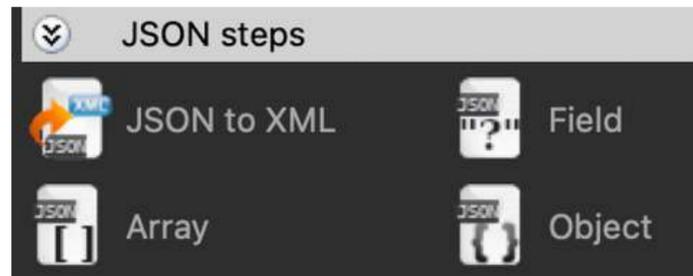
After updating the schema, an **object node** appears in the source picker



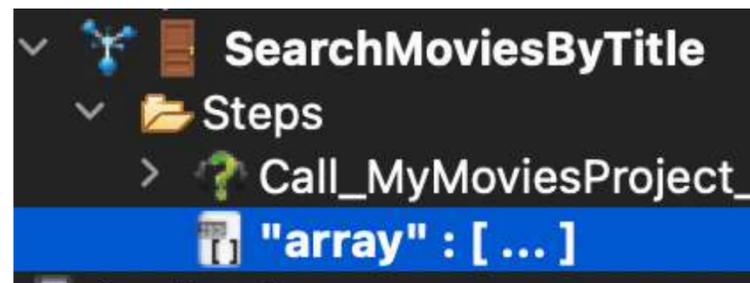
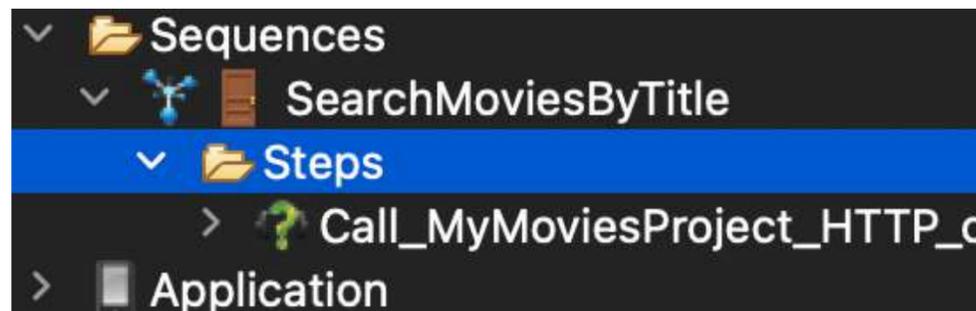
# 4.7 Create a custom data structure

Your sequence is now calling the **transaction SearchMoviesByTitle** which is a **JSON HTTP Transaction**.

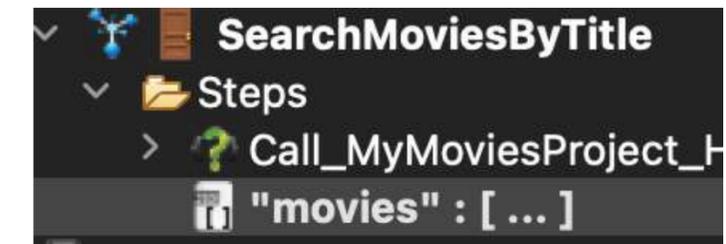
To construct our own **response data structure** from the transaction's **response data**, let's use the **Array Step (JSON step)**.



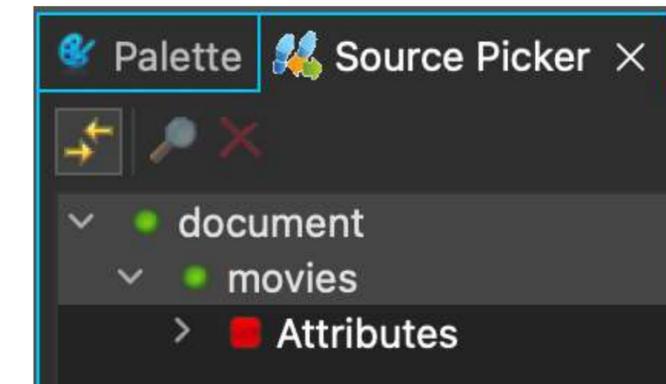
Drag the **Array step** from the palette and drop it into the **steps folder of your sequence** after the **transaction call**.



Rename it **movies**.

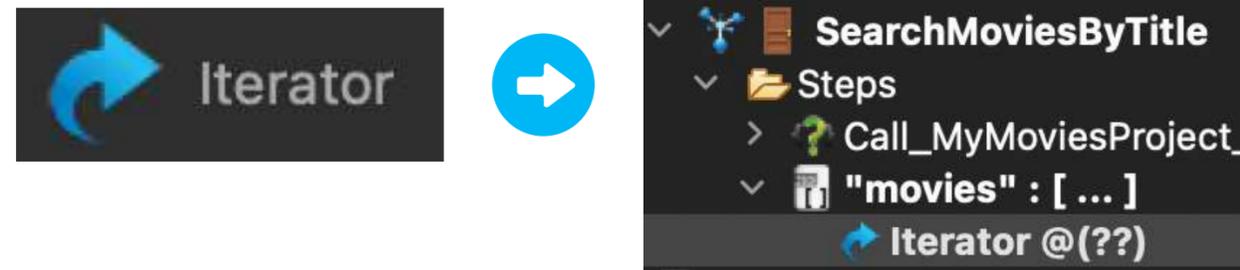


It appears as **movies** in the source picker.

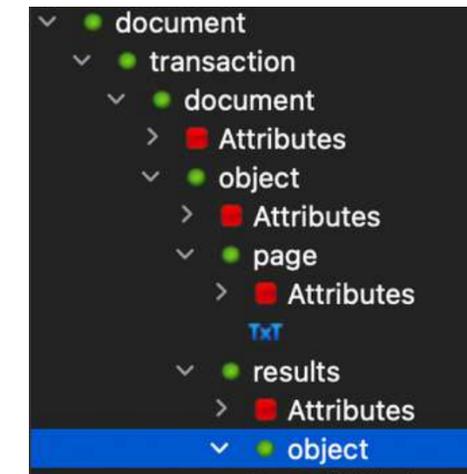


# 4.7 Create a custom data structure

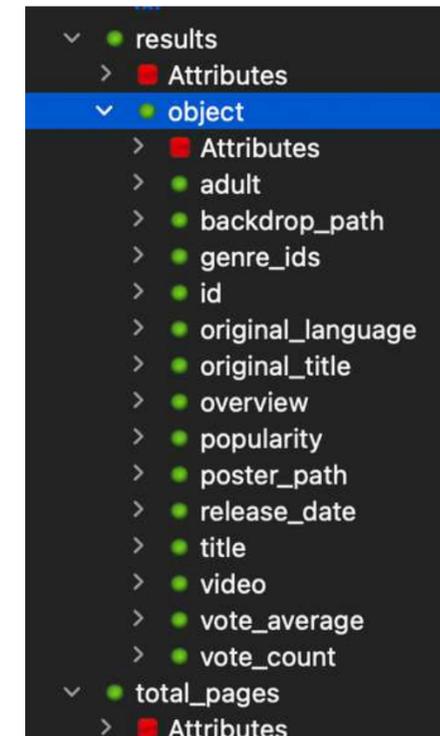
Drag the **Iterator** step from the palette and drop it into the step **movies** in your sequence.



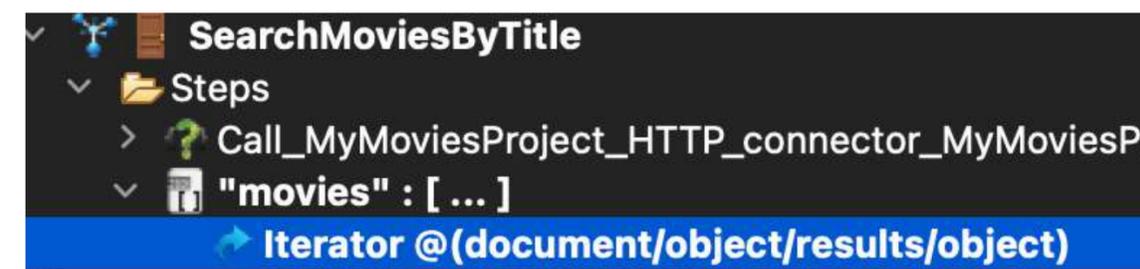
Then, double-click on the transaction call in your sequence to open the source picker.



In the source picker, expand the **results node**.

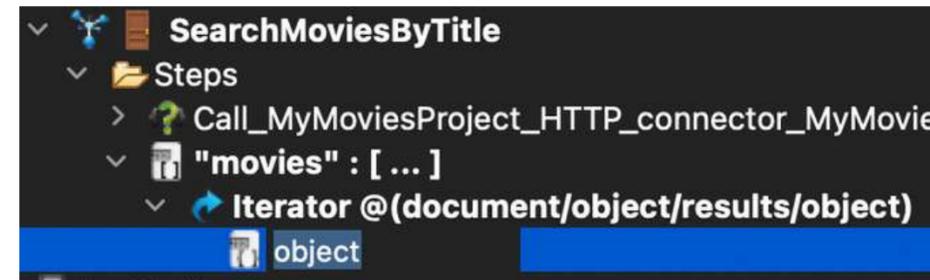
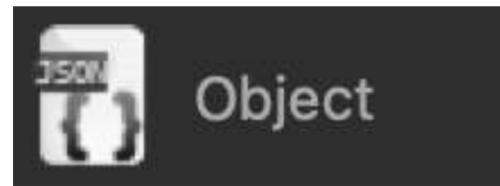


Then drag and drop the **object node** directly into your iterator. This **object node** provides the information you want in your iterator.

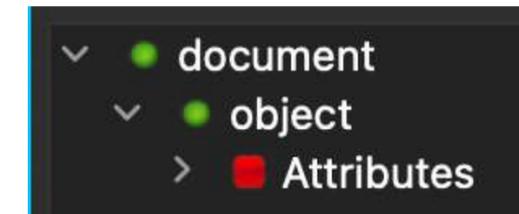


# 4.7 Create a custom data structure

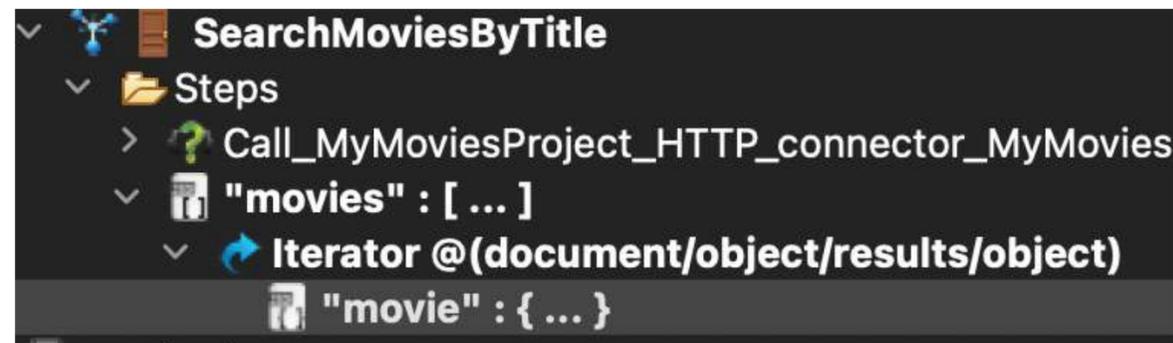
Drag the **Object** step from the palette and drop it into the **Iterator** step in your sequence.



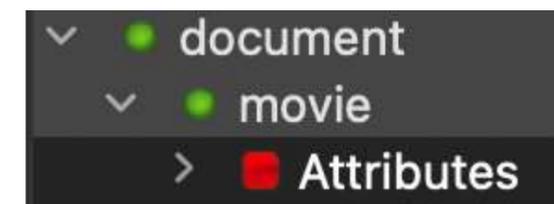
In the source picker, it appears as **object**.



The Object step is a container for the various elements you'll add to it. Let's rename it **movie** in the treeview.



Now, in the source picker, it appears as **movie**.



# 4.7 Create a custom data structure

In the **response data** from the transaction,

for each item, we receive an object movie with many fields, as shown in the source picker.

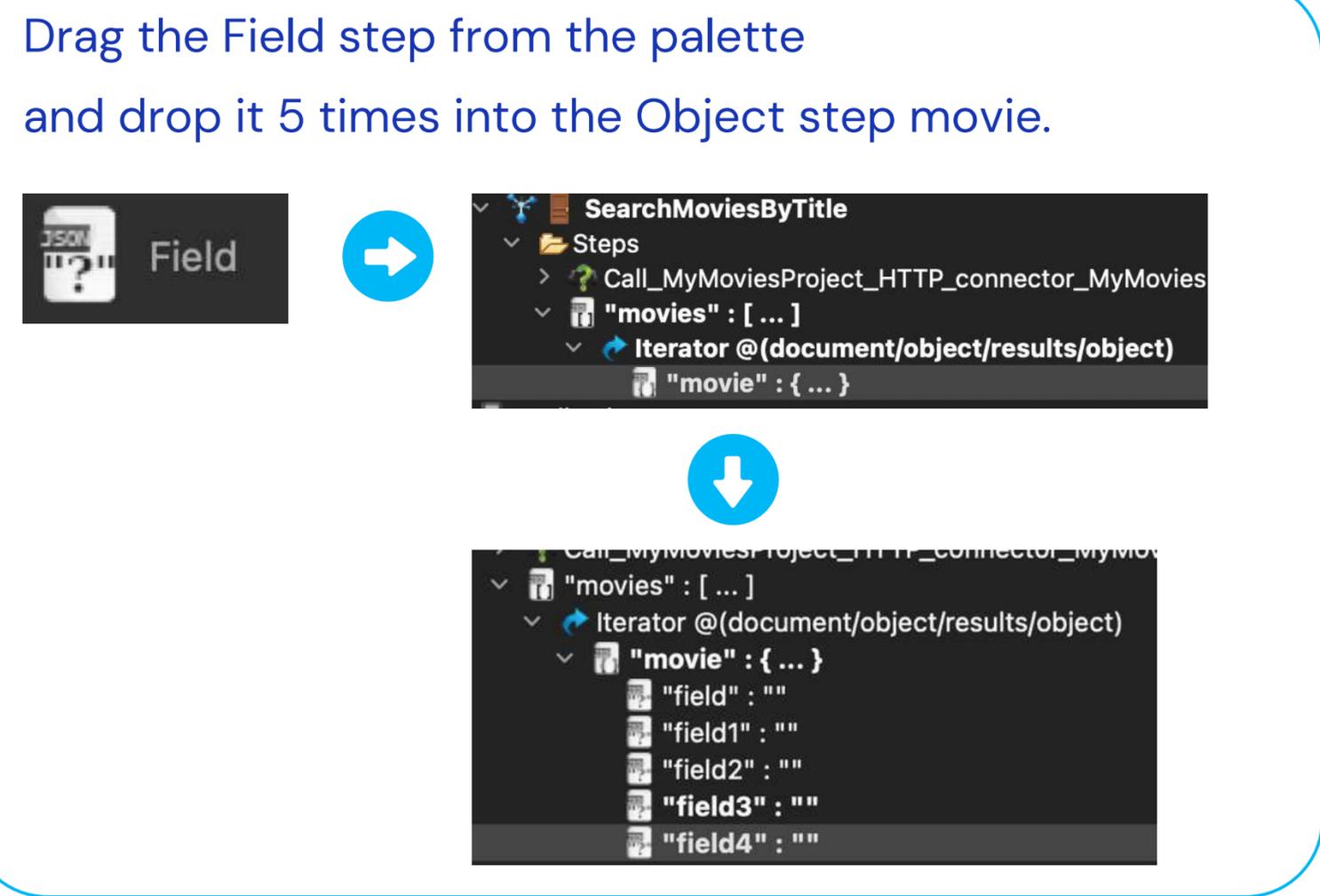
In our application, we only need a few of them and we're going to select the fields that interest us.



Let's say I want the following fields displayed in the front-end :

- title
- overview
- poster\_path
- release\_date
- original\_title

Drag the Field step from the palette and drop it 5 times into the Object step movie.



```

    SearchMoviesByTitle
    - Steps
      - Call_MyMoviesProject_HTTP_connector_MyMovies
      - "movies" : [ ... ]
        - Iterator @(document/object/results/object)
          - "movie" : { ... }
            - field
            - field1
            - field2
            - field3
            - field4
  
```



# 4.7 Create a custom data structure

Click twice on the movie step to display it in the source picker.

Rename the 5 fields in movie as title, overview, poster\_path, release\_date, original\_title

```

Can_MyMovies!_object_1111_connector_MyMovies
└─ "movies" : [ ... ]
  └─ Iterator @(document/object/results/object)
    └─ "movie" : { ... }
      └─ "field" : ""
        └─ "field1" : ""
          └─ "field2" : ""
            └─ "field3" : ""
              └─ "field4" : ""
  
```



```

Iterator @(document/object/r
└─ "movie" : { ... }
  └─ "title" : ""
    └─ "overview" : ""
      └─ "poster_path" : ""
        └─ "release_date" : ""
          └─ "original_title" : ""
  
```

In the source picker, you can see that the 5 fields in movie have been renamed as well.

```

"movie" : { ... }
  
```

```

document
└─ movie
  └─ Attributes
    └─ field
      └─ Attributes
        └─ Txt
      └─ field1
        └─ Attributes
          └─ Txt
        └─ field2
          └─ Attributes
            └─ Txt
          └─ field3
            └─ Attributes
              └─ Txt
            └─ field4
              └─ Attributes
                └─ Txt
  
```



```

document
└─ movie
  └─ Attributes
    └─ title
      └─ Attributes
        └─ Txt
      └─ overview
        └─ Attributes
          └─ Txt
        └─ poster_path
          └─ Attributes
            └─ Txt
          └─ release_date
            └─ Attributes
              └─ Txt
            └─ original_title
              └─ Attributes
                └─ Txt
  
```

The structure of each field has been renamed

```

"field" : ""
  
```

```

document
└─ field
  └─ Attributes
    └─ Txt
  
```



```

"title" : ""
  
```

```

document
└─ title
  └─ Attributes
    └─ Txt
  
```

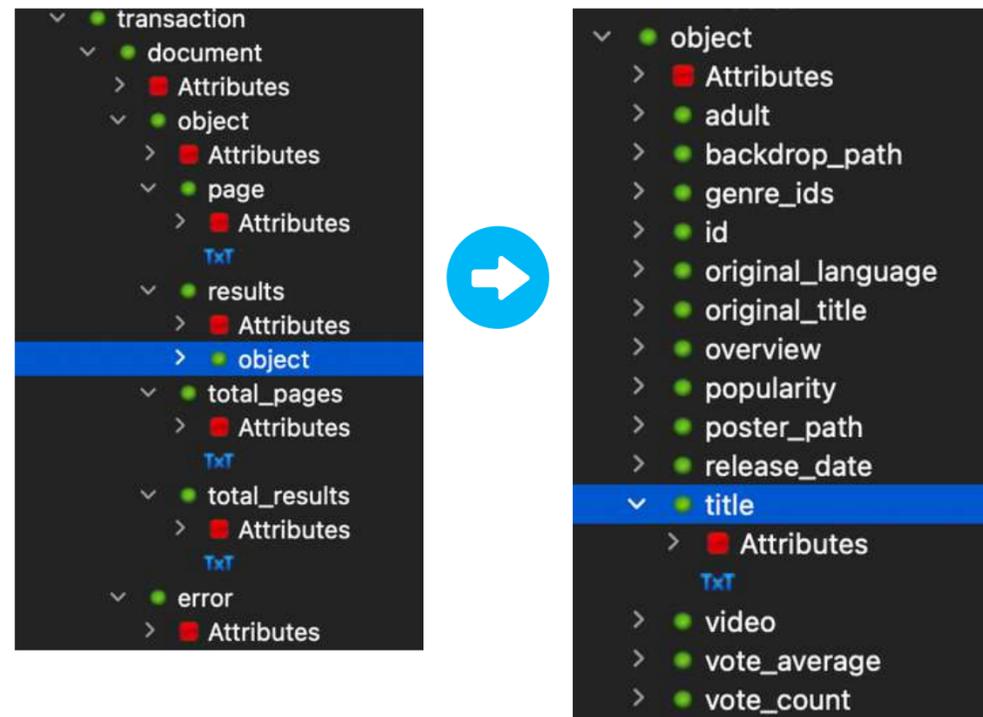


# 4.7 Create a custom data structure

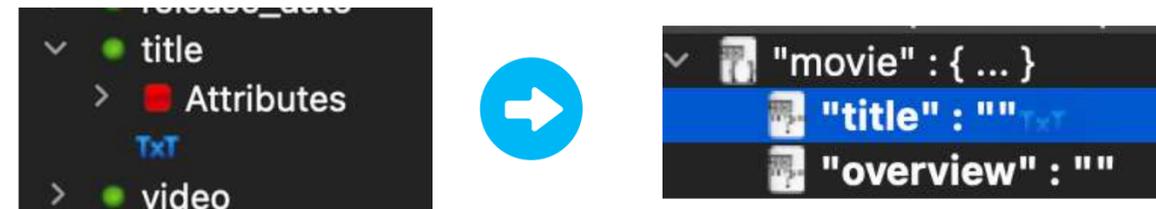
Now we want to bind these fields to the values of the fields in the iterator.

Double-click on the **Iterator step** to display its data in the Source Picker and open the object node.

```
Iterator @(document/object/results/object)
```




Drag and drop the **TxT element** corresponding to the required information into the various steps of the element.



Choose **Value** each time you are prompted to set the value property of the step.





In properties, the value appears as binded.

Property	Value
Base properties	
Comment	
Is active	true
Key	title
Type	string
Value	@[1698686296200, ./title/text()]



# 4.7 Create a custom data structure

Repeat the same operation for the 5 fields.

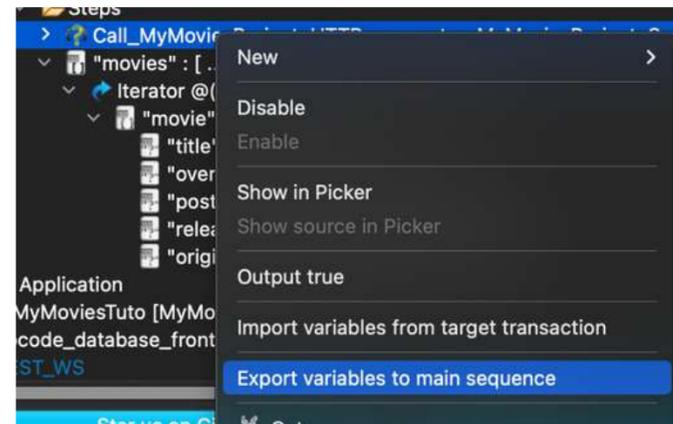
```
"movie" : { ... }
  "title" : @(title/text())
  "overview" : @(overview/text())
  "poster_path" : @(poster_path/text())
  "release_date" : @(release_date/text())
  "original_title" : @(original_title/text())
```



```
SearchMoviesByTitle
└─ Steps
   └─ Call_MyMoviesProject_HTTP_connector_MyMoviesProject_SearchMoviesByTitle
      └─ "movies" : [ ... ]
         └─ Iterator @(document/object/results/object)
            └─ "movie" : { ... }
               "title" : @(title/text())
               "overview" : @(overview/text())
               "poster_path" : @(poster_path/text())
               "release_date" : @(release_date/text())
               "original_title" : @(original_title/text())
```

Now we want to import the variables of the transaction into the sequence,

Right-click on the transaction call, and select **Export variables to main sequences.**



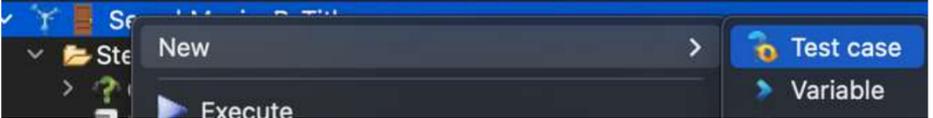
```
SearchMoviesByTitle
└─ Steps
   └─ Call_MyMoviesProject_HTTP_connector_MyMoviesProject_SearchMoviesByTitle
      └─ "movies" : [ ... ]
         └─ Iterator @(document/object/results/object)
            └─ "movie" : { ... }
               "title" : @(title/text())
               "overview" : @(overview/text())
               "poster_path" : @(poster_path/text())
               "release_date" : @(release_date/text())
               "original_title" : @(original_title/text())
          └─ Variables
              └─ __header_Authorization = "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIyYjZkdjE1MjE1YyJ9"
                 movieTitle
```

A folder **Variables** has been added to the sequence.

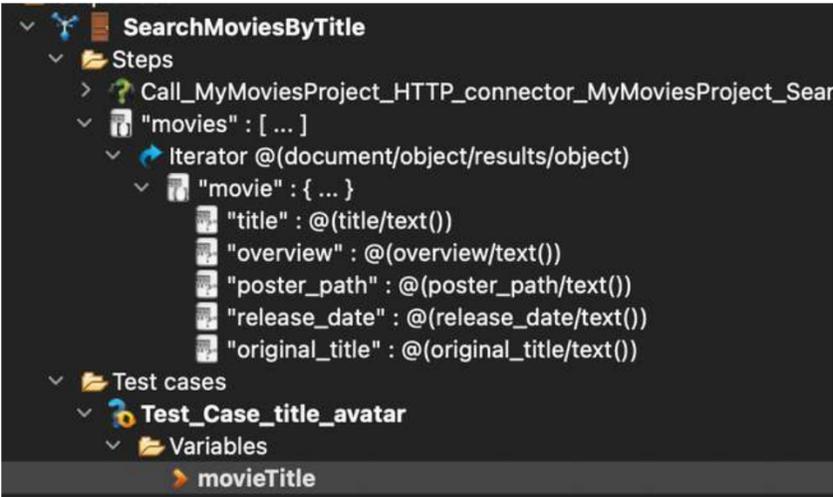


# 4.8 Test the sequence

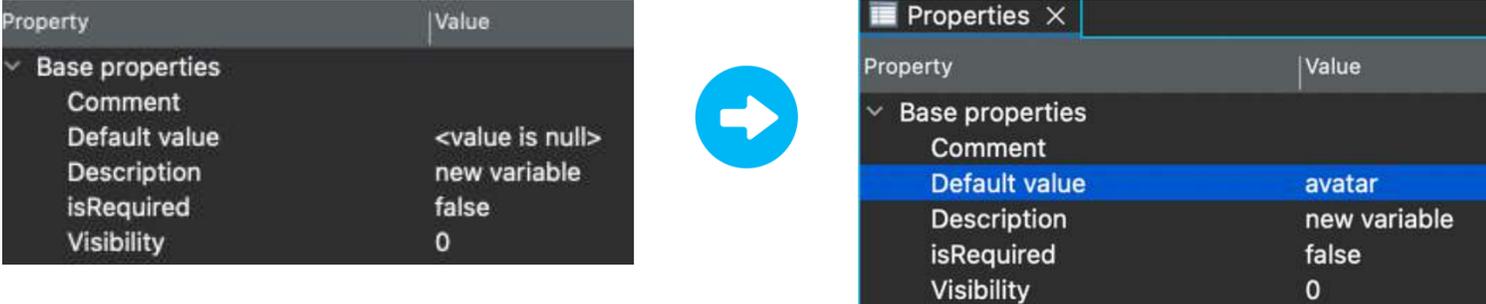
Now, let's create a test case for the sequence  
(as shown in the previous slides for the transaction SearchMoviesByTitle).



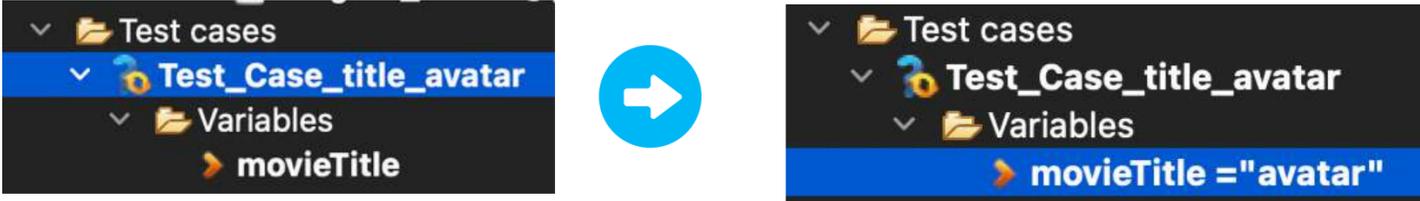
↓



Click on the variable movieTitle in the test case.  
In properties, change the Default value of movieTitle to "avatar".

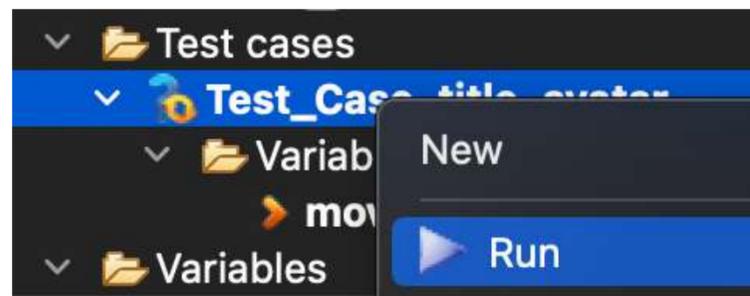


The value appears in the treeview.

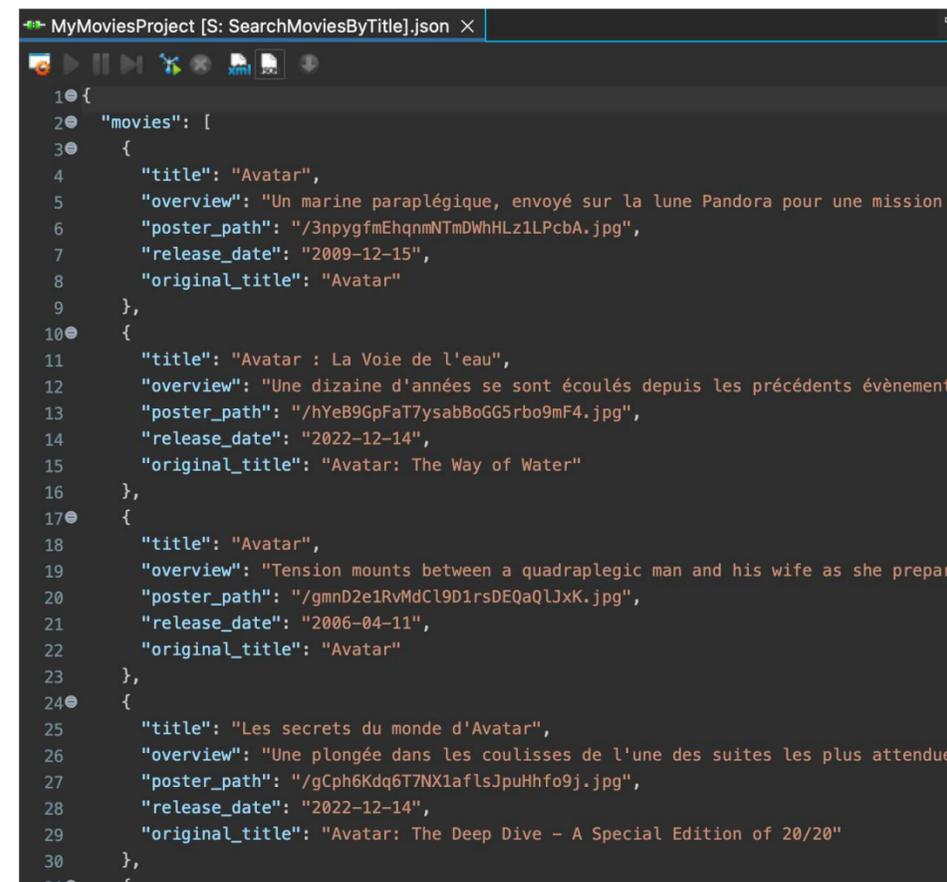



# 4.8 Test the sequence

Right-click on the test case, choose **Run** to execute it and generate response data.

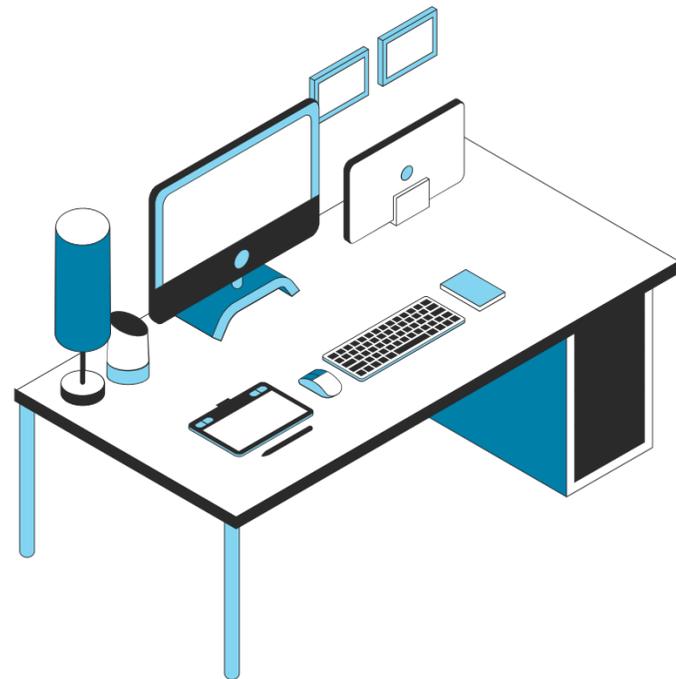


The results are displayed in the editors panel. The response data generated by the sequence will display only the information you requested.



# 5 – JavaScript Scope

How to handle JavaScript in the studio.



**6.1** What is the JavaScript Scope ?

---

**6.2** Interactions with JS Scope

---

**6.3** Back-end Objects bound to JS Scope

---

**6.4** Step Sequence JS

---

**6.5** Step Input variables

---

**6.6** Modify a sequence with the JS Scope

# 5.1 What is the JavaScript Scope ?

By default, **every execution** of a transaction or a sequence has a **JavaScript environment**.

This is called the **JavaScript Scope**.

You can use JS to **manipulate data** in the sequence.

For example, perform calculations and data transformations...

## Transaction or sequence input variables

All variables declared as **input vars** (input variables) of the sequence

- are inserted into the global scope of the JS environment.
- are automatically JavaScript variables
- become global variables of the sequence.



## 5.2 Interactions with JS Scope

In order to **manipulate data in JavaScript**,

Convertigo uses **backend objects** as gateways between the **structured context** and the **JS scope**.

These objects

- manage **interactions** between XML data sources and JavaScript.
- are used as **steps in sequences**.

These objects or steps can either

- **transform XML data** from the source defined in the Source property **into JavaScript variables** in the current executed sequence JS scope.

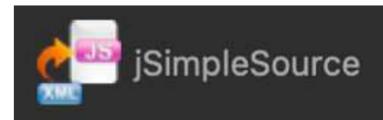
These JS variables can be manipulated in JS.

- **transform JavaScript scope variables** into **XML data sources**.
- use **JavaScript expressions** as **data sources**.



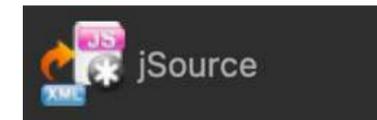
## 5.3 Back-end Objects bound to JS Scope

Steps transforming XML data sources into JavaScript variables



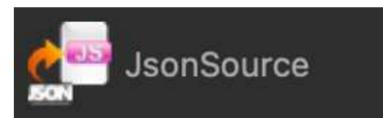
### jSimpleSource – JS step

This step **transforms a single node** from the source defined in the Source property into a **JS variable** (String)



### jSource – JS step

This step **transforms a list of XML nodes** into a **JS variable** (Java NodeList object)



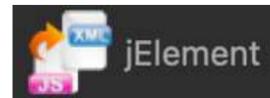
### JsonSource – JS step

This step **extracts a JSON typed XML structure** from the source defined in the Source property, parses it as JSON, and sets it as a **JS variable** (JS Object or JS Array).



## 5.3 Back-end Objects bound to JS Scope

Steps transforming JS variables into XML



### jElement – XML Step

This step **adds an XML element node** based on a **JS expression** to parent XML element in the **sequence XML output**.



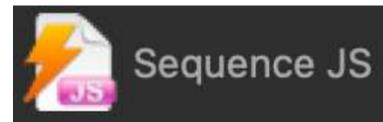
### JSON to XML – JSON step

This step **adds an XML attribute node** based on a **JS expression** to parent XML element in the **sequence XML output**.



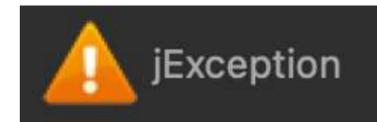
# 5.3 Back-end Objects bound to JS Scope

## Steps used to manipulate JavaScript



### Sequence JS – JS step

This step is used to **write JavaScript code** which is **executed in the sequence scope** (initialize variables, calculations...)



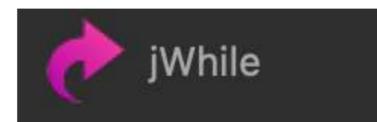
### jException – JS step

This step **raises a Convertigo Engine exception**. It **breaks the sequence execution flow**, ending the sequence just after this step.



### jIf – Flow control step

This step is **based on a JavaScript condition** and contains other steps executed **only if the condition is fulfilled**.



### jWhile – Flow control step

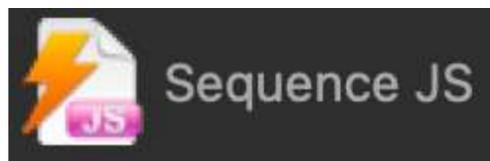
This step **executes a group of child steps** as the **condition expression** set in the Condition property remains true.



## 5.4 Sequence JS Step

The JS Scope is useful to **modify Sequences**.

When you need to **write code directly in JavaScript**, the **Sequence JS step** is very helpful.  
This JavaScript code will be **executed in the sequence scope**.

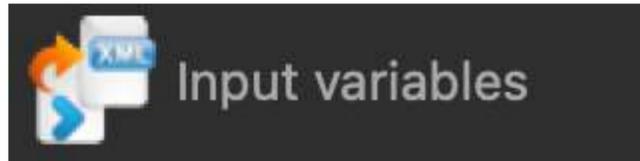


With the Sequence JS step, you can :

- initialize variables,
- perform complex calculations,
- access the context object to get useful properties  
(contextID, httpSession, isCacheEnabled, lockPooledContext, etc.)
- use some context methods to manipulate the result XML DOM,  
encode and decode data, abort sequence...

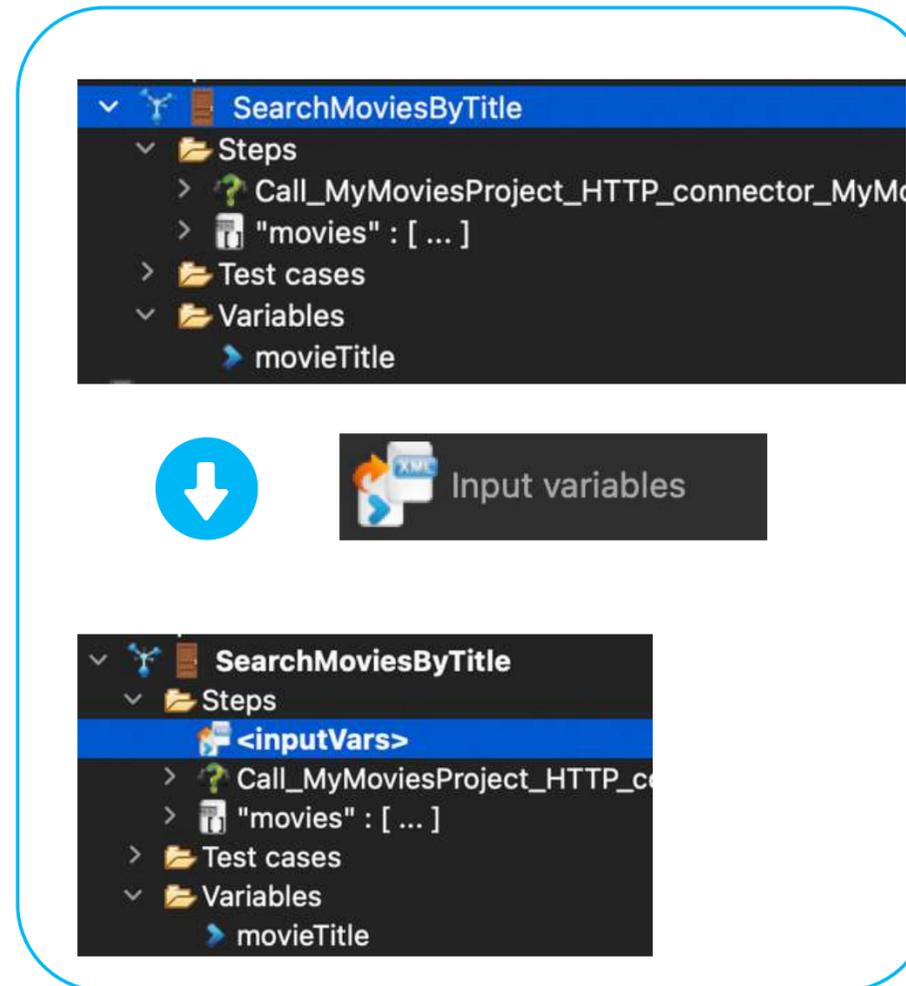


# 5.5 Input variables Step

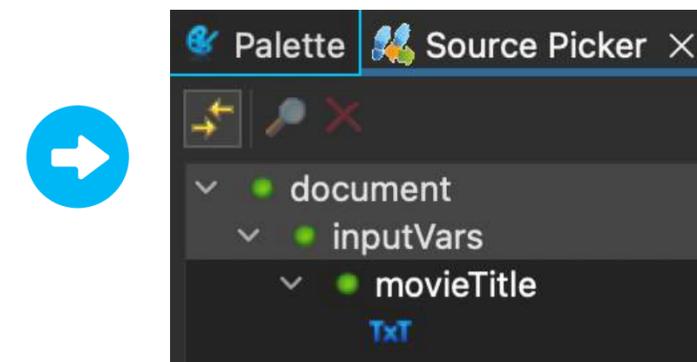


The **step Input variables** is an XML element containing dynamically the **input variables of parent Sequence**.

Placed at the **beginning of a Sequence**, this step allows **steps ordered after** to **use the Sequence input variables as source**.



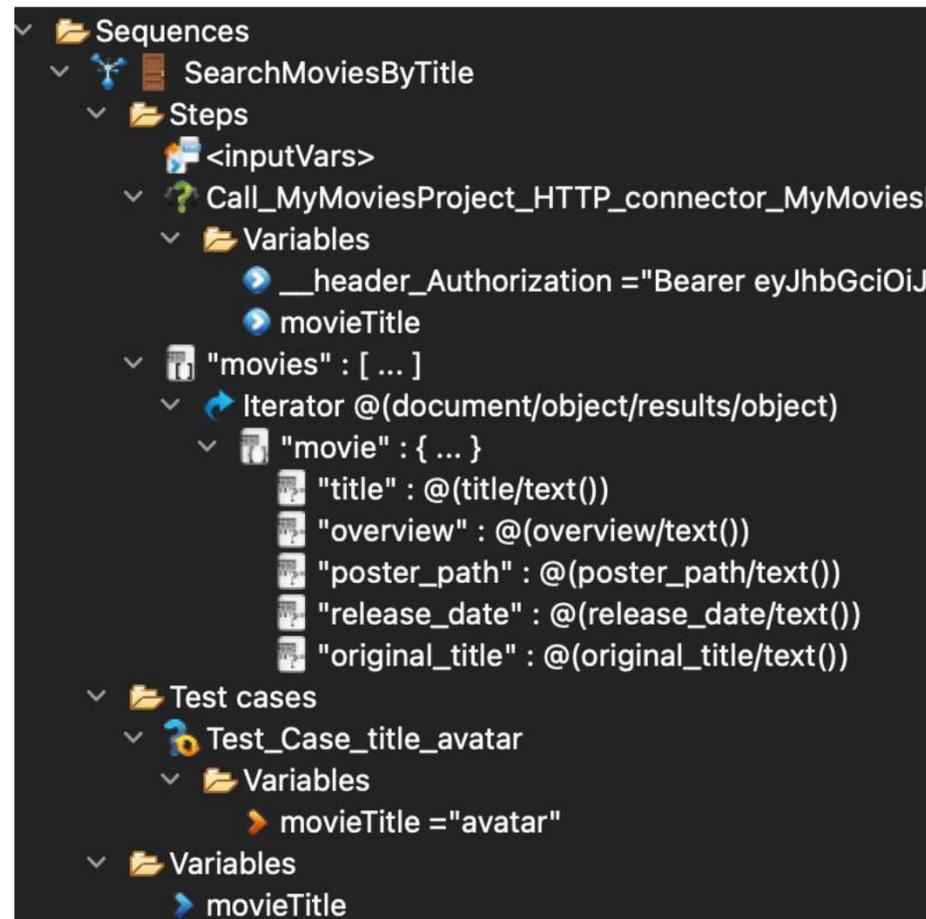
When you add it as the first step of the sequence, it **appears as a source** in the **source picker**.



# 5.6 Modify a sequence with the JS Scope

## Exercice 1 : Change a variable name with Sequence JS

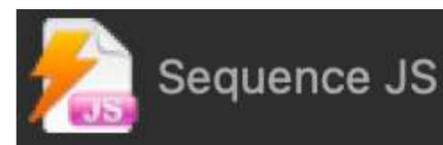
Here is our sequence SearchMoviesByTitle



Let's say we want the name of the **input variable "movieTitle"** to appear as **title** in our sequence.



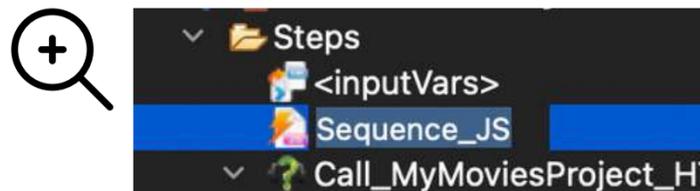
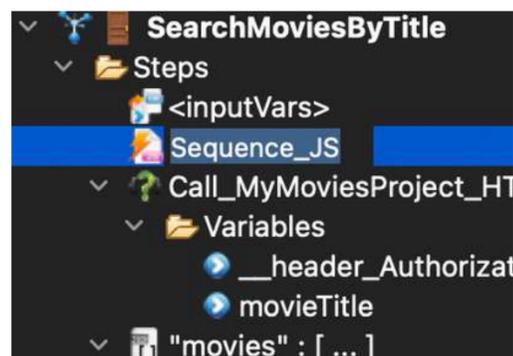
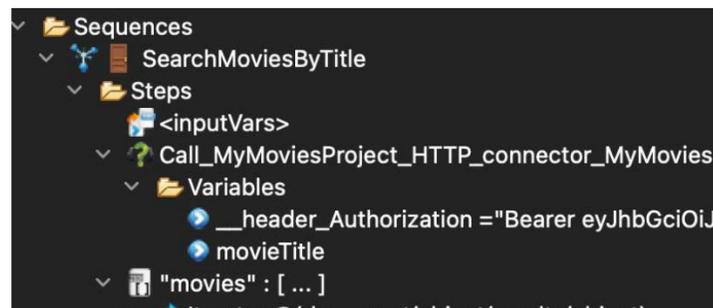
To change the name of the input variable "movieTitle", we are going to use JavaScript in a **Sequence JS**.



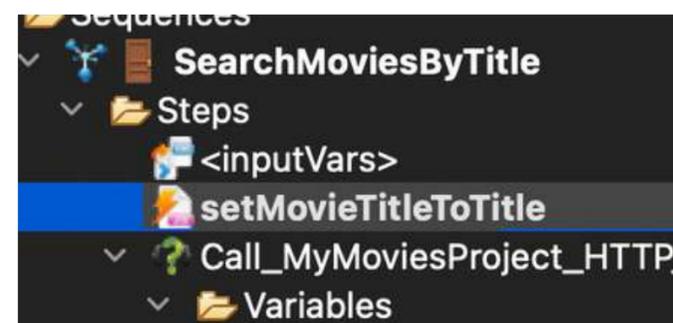
# 5.6 Modify a sequence with the JS Scope

## Exercice 1 : Change a variable name with Sequence JS

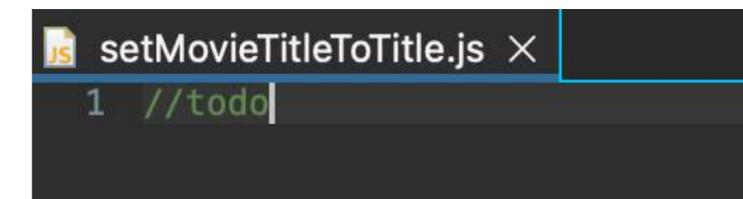
Drag the **Sequence JS** step from the palette in the steps folder after the step InputVars.



Rename it setMovieTitleToTitle.



Click twice on **setMovieTitleToTitle** to open the **file setMovieTitleToTitle.js** in the editor panel

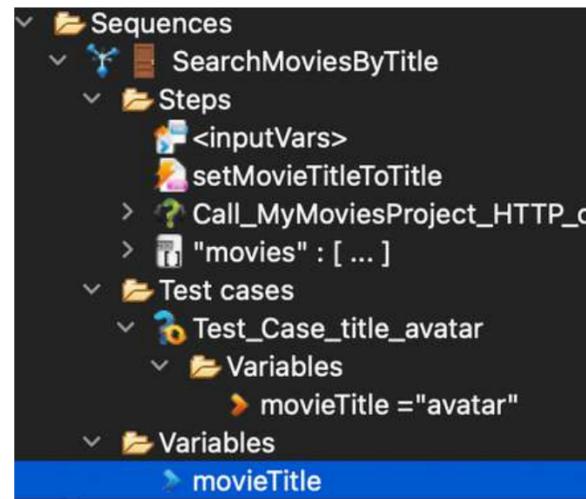


Change the variable name in the file with JavaScript.

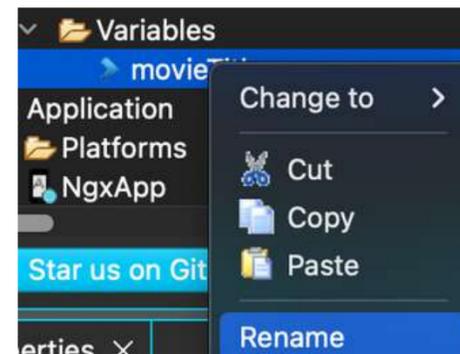


# 5.6 Modify a sequence with the JS Scope

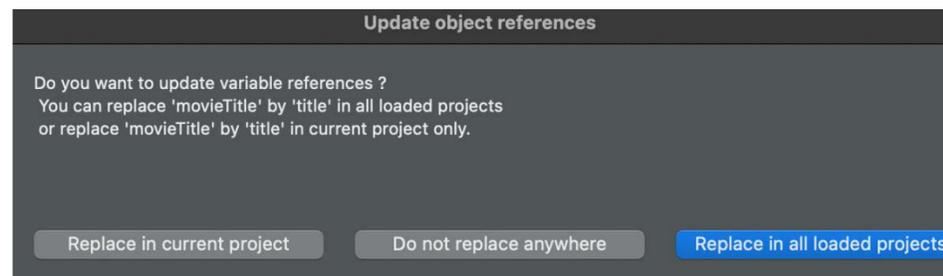
## Exercise 1: Change a variable name with Sequence JS



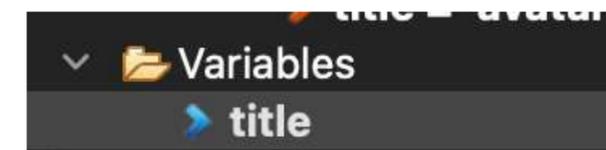
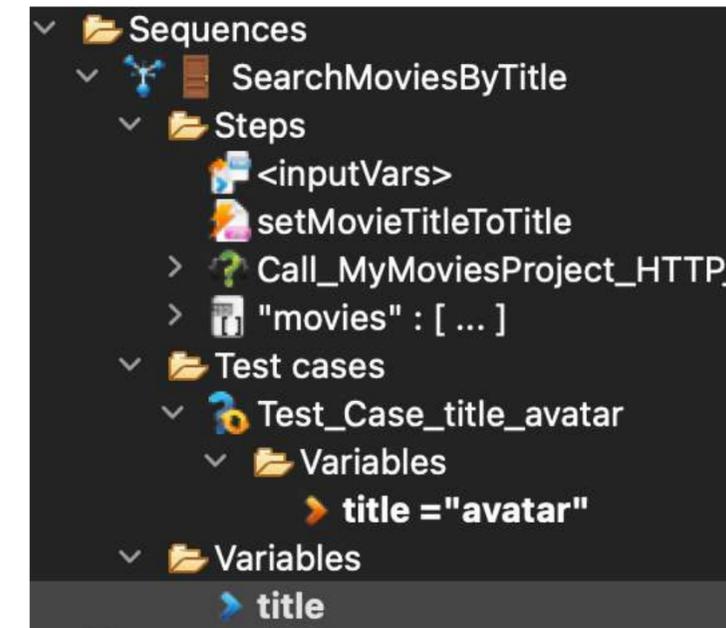
Rename the **movieTitle** variable to **title** in the **Variables** folder of the sequence.



In the **Update object references** window, select **Replace in current project**.



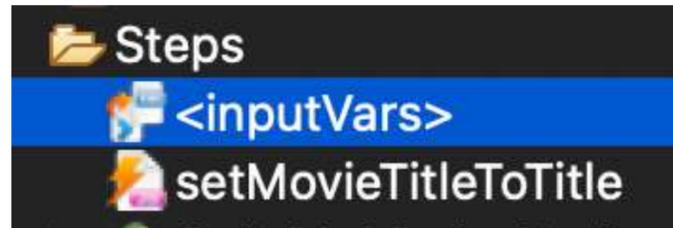
The variable **appears as title** in the sequence (Test cases, Variables folder)



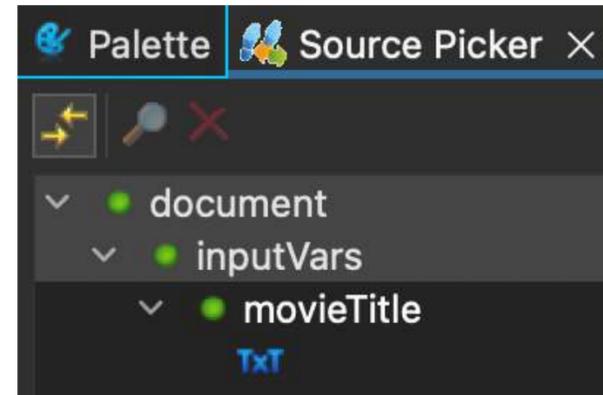
## 5.6 Modify a sequence with the JS Scope

### Exercice 1 : Change a variable name with Sequence JS

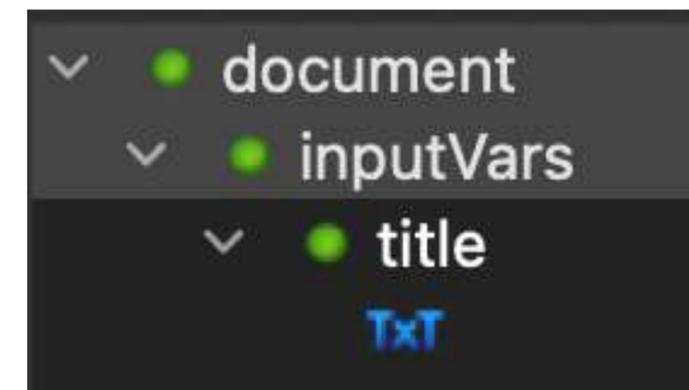
When focused on **inputVars** step,  
the source picker shows the entry variable of the sequence as **title**  
(not as movieTitle anymore).



Before



Now



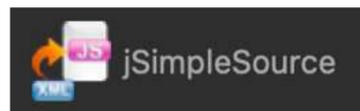
# 5.6 Modify a sequence with the JS Scope

## Exercice 2 : Set the title to uppercase with Sequence JS

Let's say we want a field with the title in uppercase in our JSON data.

```

Steps
├── <inputVars>
├── setMovieTitleToTitle
├── Call_MyMoviesProject_HTTP_connector_MyMov
├── "movies" : [ ... ]
└── Iterator @(document/object/results/object)
    ├── "movie" : { ... }
    │   ├── "title" : @(title/text())
    │   ├── "overview" : @(overview/text())
    │   ├── "poster_path" : @(poster_path/text())
    │   ├── "release_date" : @(release_date/text())
    │   └── "original_title" : @(original_title/text())
    
```



In the movie object, after the field steps, we add a **jSimpleSource** step..

```

"movie" : { ... }
├── "title" : @(title/text())
├── "overview" : @(overview/text())
├── "poster_path" : @(poster_path/text())
├── "release_date" : @(release_date/text())
├── "original_title" : @(original_title/text())
└── myVariable
    
```



We name it **jUpperCaseTitle** (for JS variables, good practice is to add a "j" at the beginning)

```

"movie" : { ... }
├── "title" : @(title/text())
├── "overview" : @(overview/text())
├── "poster_path" : @(poster_path/text())
├── "release_date" : @(release_date/text())
├── "original_title" : @(original_title/text())
└── jUpperCaseTitle @(??)
    
```



# 5.6 Modify a sequence with the JS Scope

## Exercice 2 : Set the title to uppercase with Sequence JS

The jSimpleSource step is used to transform a single node from a source into a JS variable. Now we want to bind it to the value of the field title in the iterator.

Double-click on the **Iterator step** to display its data in the Source Picker and open the object node.

```
"movies" : [ ... ]
  Iterator @(document/object/results/
    "movie" : { ... }
      "title" : @(title/text())
      "overview" : @(overview/text())
      "poster_path" : @(poster_path/text())
      "release_date" : @(release_date/text())
      "original_title" : @(original_title/text())
      jUpperCaseTitle @(??)
```



```
document
  transaction
    document
      Attributes
      object
        Attributes
        page
          Attributes
          results
            Attributes
            object
            total_pages
              Attributes
            total_results
              Attributes
            error
              Attributes
            code
```



```
object
  Attributes
  adult
  backdrop_path
  genre_ids
  id
  original_language
  original_title
  overview
  popularity
  poster_path
  release_date
  title
  video
  vote_average
  vote_count
```



Drag and drop the **TxT** element of the title into the jSimpleSource step.



# 5.6 Modify a sequence with the JS Scope

## Exercice 2 : Set the title to uppercase with Sequence JS

Add a Sequence JS step in the sequence, after the step jUpperCaseTitle.

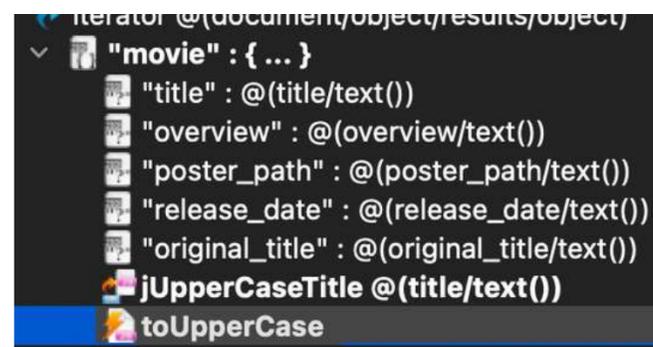


```

"movies" : [ ... ]
  Iterator @(document/object/results/object)
    "movie" : { ... }
      "title" : @(title/text())
      "overview" : @(overview/text())
      "poster_path" : @(poster_path/text())
      "release_date" : @(release_date/text())
      "original_title" : @(original_title/text())
      jUpperCaseTitle @(title/text())
    Sequence_JS
  
```



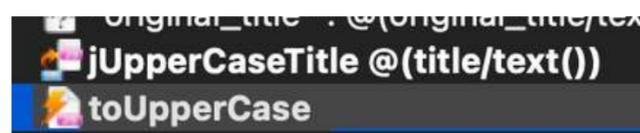
Rename it toUpperCase.



```

"movie" : { ... }
  "title" : @(title/text())
  "overview" : @(overview/text())
  "poster_path" : @(poster_path/text())
  "release_date" : @(release_date/text())
  "original_title" : @(original_title/text())
  jUpperCaseTitle @(title/text())
  toUpperCase

```

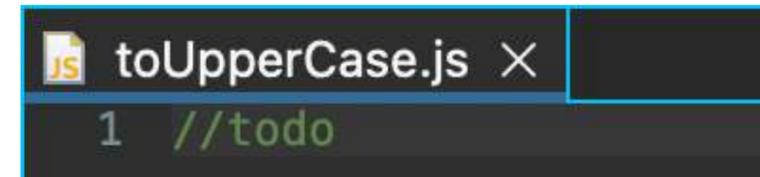
```

original_title : @(original_title/text())
jUpperCaseTitle @(title/text())
toUpperCase

```



Click twice on the step to open toUpperCase.js



```

toUpperCase.js X
1 //todo

```



Edit the toUpperCase.js with JS code.



```

toUpperCase.js X
1 jUpperCaseTitle = jUpperCaseTitle.toUpperCase();

```



# 5.6 Modify a sequence with the JS Scope

## Exercice 2 : Set the title to uppercase with Sequence JS

Add a field step after toUpperCase and name it **upperCaseTitle**.

```

    iterator @(document/object/results/object)
  > "movie" : { ... }
    "title" : @(title/text())
    "overview" : @(overview/text())
    "poster_path" : @(poster_path/text())
    "release_date" : @(release_date/text())
    "original_title" : @(original_title/text())
    jUpperCaseTitle @(title/text())
    toUpperCase
  field
  
```



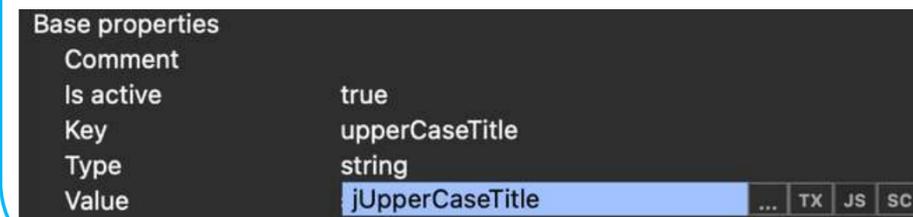
```

    "movie" : { ... }
    "title" : @(title/text())
    "overview" : @(overview/text())
    "poster_path" : @(poster_path/text())
    "release_date" : @(release_date/text())
    "original_title" : @(original_title/text())
    jUpperCaseTitle @(title/text())
    toUpperCase
    "upperCaseTitle" : ""
  
```

In the **Value** property of upperCaseTitle, select the **JS Scope** by clicking on JS.



Enter **jUpperCaseTitle** to select the JS variable as value.



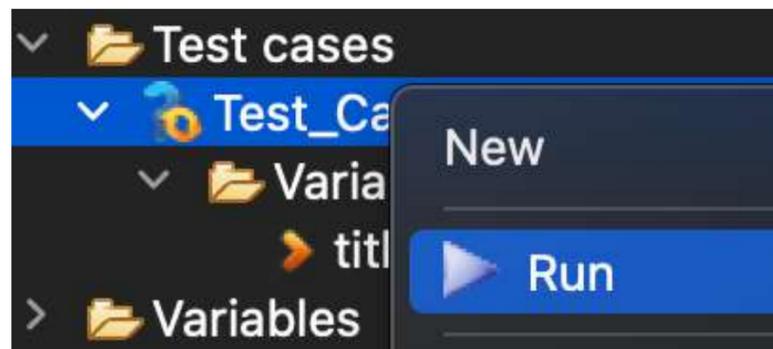
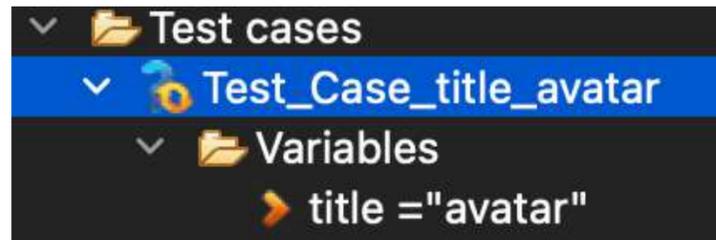
The value of upperCaseTitle is now sourced on the value of the JS variable jUpperCaseTitle.



# 5.6 Modify a sequence with the JS Scope

## Exercice 2 : Set the title to uppercase with Sequence JS

Let's run the Test Case



The key upperCaseTitle and the value in upperCase appears in the response data

```

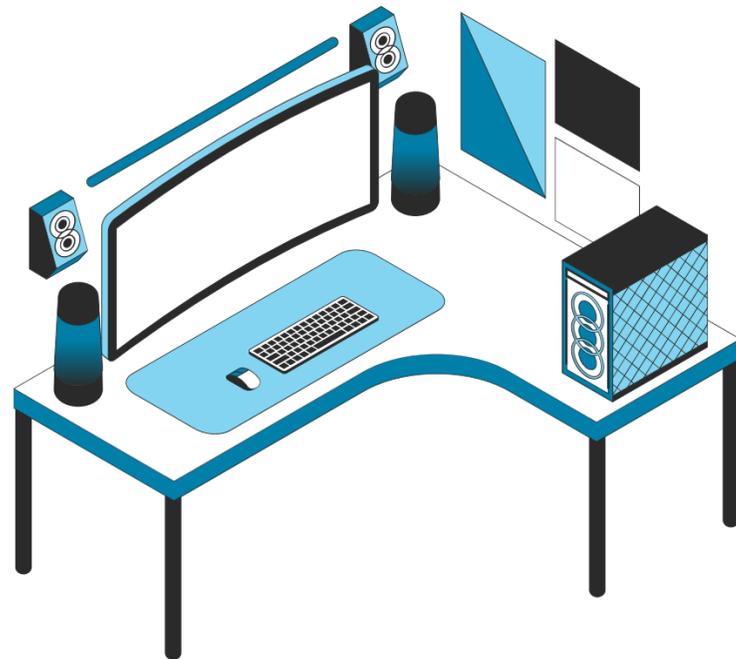
1 | {
2 |   "movies": [
3 |     {
4 |       "title": "Avatar",
5 |       "overview": "Un marine paraplégique, envoyé sur la lune Pandora",
6 |       "poster_path": "/3npygfmEhqnmNTmDWhHLz1LPcbA.jpg",
7 |       "release_date": "2009-12-15",
8 |       "original_title": "Avatar",
9 |       "upperCaseTitle": "AVATAR"
10 |    },
11 |    {
12 |      "title": "Avatar : La Voie de l'eau",
13 |      "overview": "Une dizaine d'années se sont écoulés depuis les pr",
14 |      "poster_path": "/hYeB9GpFaT7ysabBoGG5rbo9mF4.jpg",
15 |      "release_date": "2022-12-14",
16 |      "original_title": "Avatar: The Way of Water",

```



# 6 – Error Management

How to handle errors in the studio.



**6.1** Basics on Error Management

---

**6.2** Error Management steps

---

**6.3** Error node & error tag

---

**6.4** Using the IfExist step

---

**6.5** Using the Error Structure step

---

**6.6** Using the Return step

# 6.1 Basics on Error Management

During its execution, a step can fail, and an error happens.

There are two types of errors:

- Functional errors
- System errors

In Convertigo, to handle errors:

- We don't start with "If everything is OK, then...Or else..."  
=> otherwise, there would be too much depth in the tree structure.
- We start with "If there is a problem, then...Or else...,"  
=> It means we begin error handling before dealing with successful execution.

After each transaction call in a sequence, we test for errors.



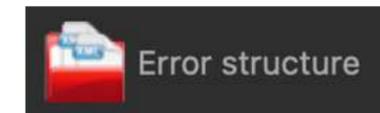
## 6.2 Error Management steps

Convertigo provides steps to handle errors in sequences.



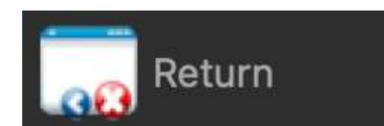
### IfExist – Flow control step

This step is used to define an **IF condition looking for node(s) on a source**.  
It contains **other steps executed only if the source** defined through the Source property **exists**.



### Error structure – XML step

This step is used  
to **generate an output XML structure**  
corresponding to an **applicative error**.  
It **doesn't break the sequence execution flow**.



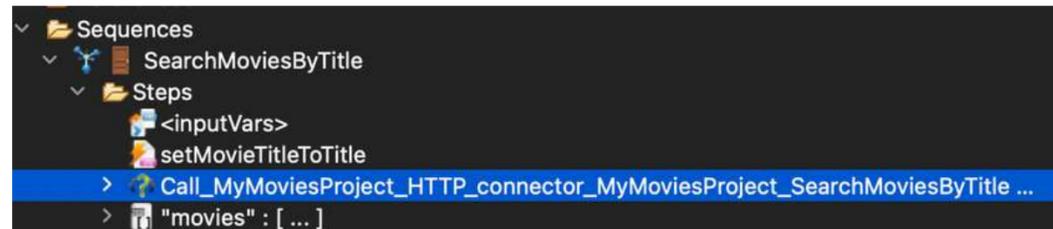
### Return – Flow control step

This step is used to **exit the current sequence**  
in which it is positioned.

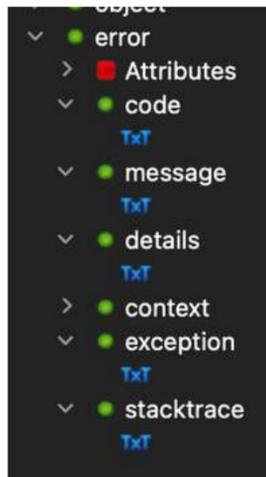
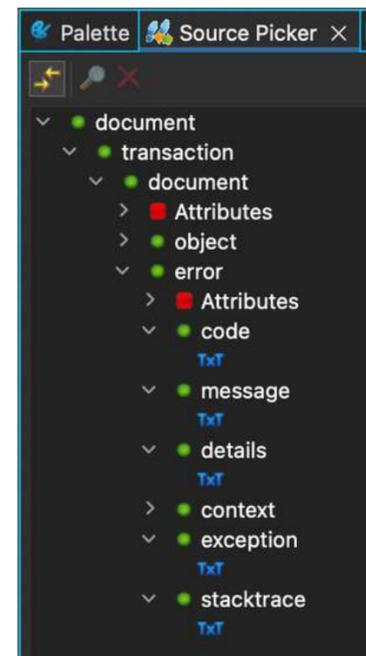


## 6.3 Error node & error tag

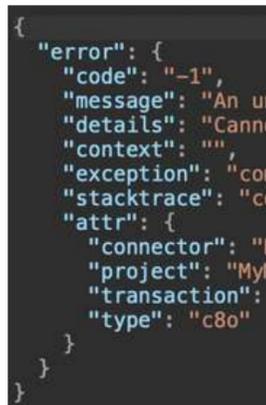
Let's have a look on the XML structure of the transaction call in the sequence SearchMoviesByTitle.



Double click on the transaction to display its structure in the source picker



In the **XML structure of a source**, there is always an **error node**, so that **errors can be picked or sourced** if they are present.



When a system error or a functional error happens, it **generates an error tag** which "fills" the error node.

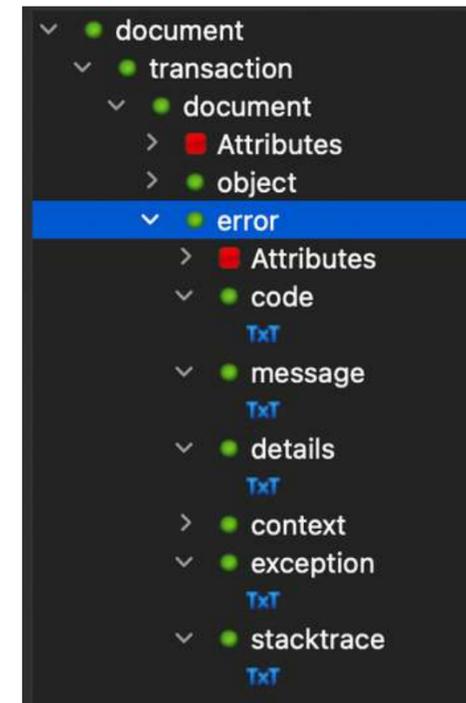
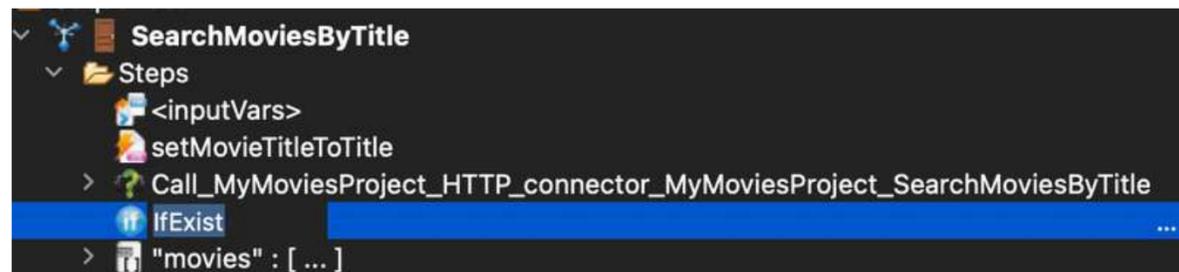
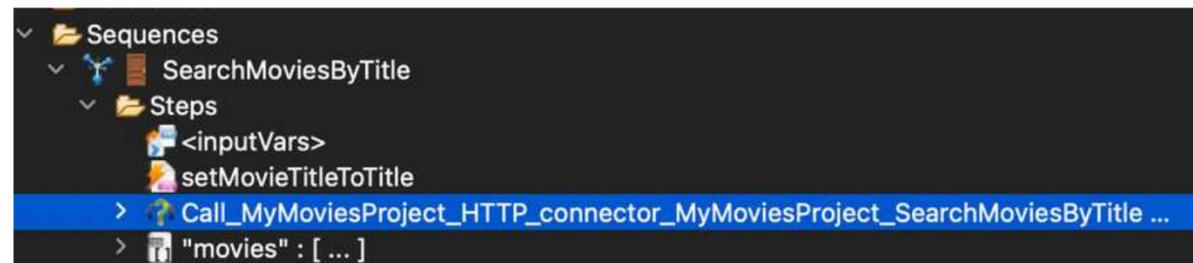
The **error tag structure is standardized**.

The error will always be in the same place and have the same format in the sequence.

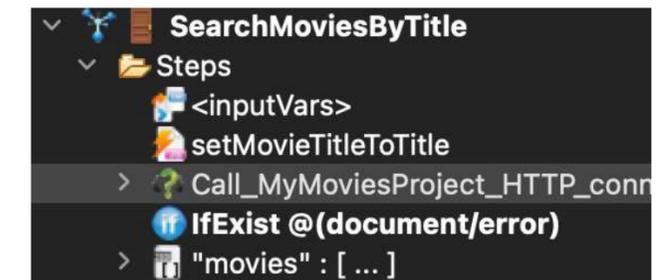


# 6.4 Using the IfExist step

In the sequence SearchMoviesByTitle, just after the transaction call and before the array movies, let's add an IfExist step.



Then drag the node error of the XML structure of the transaction call in the step IfExist.



When there is any kind of error,

an **error tag is generated within the error node.**

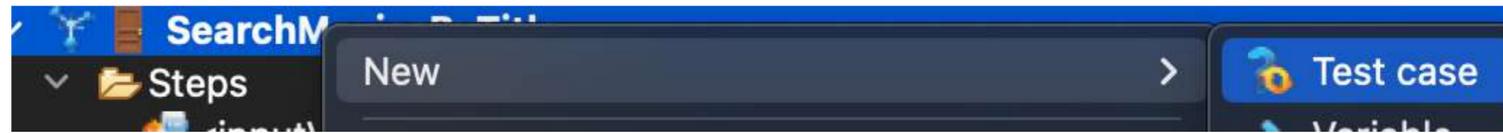
The IfExists step **checks the XPath of the transaction call**

to see if there is an **error tag in the XPath.**

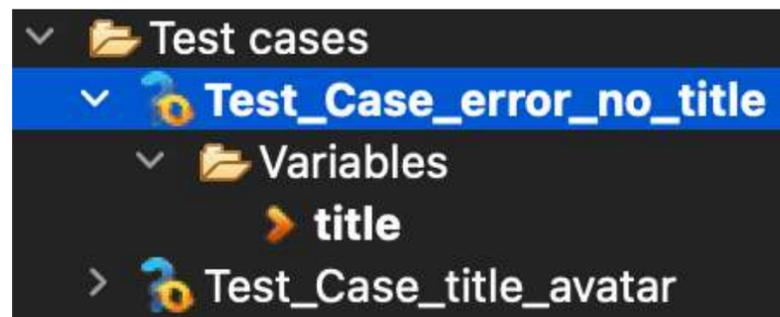


## 6.4 Using the IfExist step

Let's test it by adding a new test case with an error.



Our test case is created.



Base properties	
Comment	
Default value	<value is null>
Description	new variable
isRequired	false
Visibility	0

In the properties, the Default value of variable title is null.

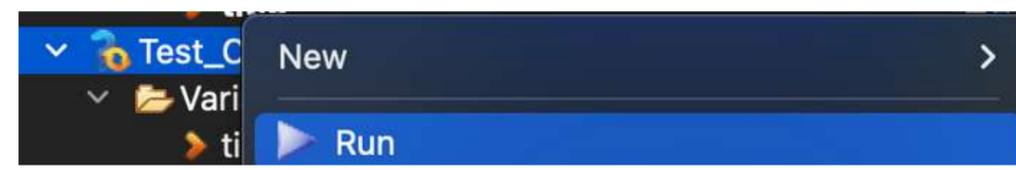
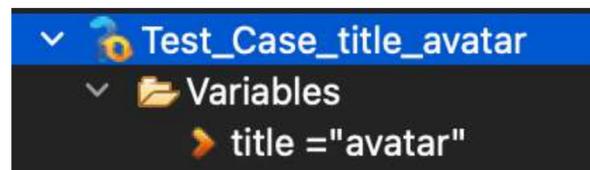
The transaction needs a title to search a movie.

If we forget to add a value to the variable title, an error will be generated.



# 6.4 Using the IfExist step

A a reminder, this is the result of the sequence execution with a test case where the title has a value and everything is OK.



```

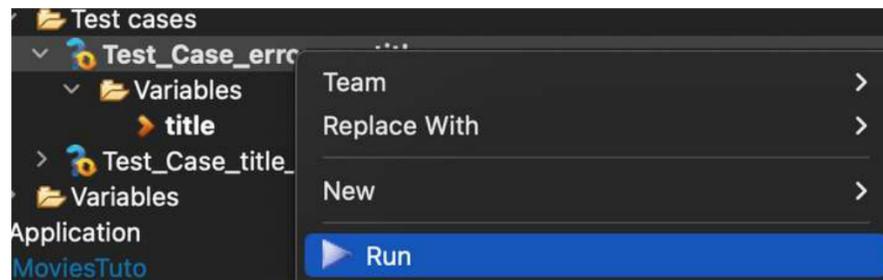
MyMoviesProject [S: SearchMoviesByTitle].json
1 {
2   "movies": [
3     {
4       "title": "Avatar",
5       "overview": "Un marine paraplégique, envoyé sur la lune Pandora pour une mission unique, est tiré en hélicoptère dans une zone dangereuse.",
6       "poster_path": "/3npygfmEhqnmNTmDWhHLz1LPcbA.jpg",
7       "release_date": "2009-12-15",
8       "original_title": "Avatar",
9       "upperCaseTitle": "AVATAR"
10    },
11   {
12     "title": "Avatar : La Voie de l'eau",
13     "overview": "Une dizaine d'années se sont écoulés depuis les précédents événements survenus sur Pandora. Jake Sully et Neytiri sont devenus parents. Leur vie idyllique, proche de la nature, est menacée lorsque la « Ressourcement Développement Administration », dangereuse organisation non-gouvernementale",
14     "poster_path": "/hYeB9GpFaT7ysabBoGG5rbo9mF4.jpg",
15     "release_date": "2022-12-14",
16     "original_title": "Avatar: The Way of Water",
17     "upperCaseTitle": "AVATAR : LA VOIE DE L'EAU"
18   },
19   {
20     "title": "Avatar",
21     "overview": "Tension mounts between a quadriplegic man and his wife as she prepares a bath for him.",
22     "poster_path": "/gmnD2e1RvMdCl9D1rsDEQaQlJxK.jpg",
23     "release_date": "2006-04-11",
24     "original_title": "Avatar",
25     "upperCaseTitle": "AVATAR"
26   },
27   {
28     "title": "Les secrets du monde d'Avatar",
29     "overview": "Une plongée dans les coulisses de l'une des suites les plus attendues du cinéma, avec des images inédites et des interviews exclusives.",
30     "poster_path": "/gCph6Kdq6T7NX1aflsJpuHhfo9j.jpg",
31     "release_date": "2022-12-14",
32     "original_title": "Avatar: The Deep Dive - A Special Edition of 20/20"
33   }
34 ]
35 }
MyMoviesProject [C: HTTP_connector_MyMoviesProject].json
1 {
2   "object": {
3     "page": 1,
4     "results": [
5       {
6         "adult": false,
7         "backdrop_path": "/vL5LR6WdxWPjLPFRLe133jXwsh5.jpg",
8         "genre_ids": [
9           28,
10          12,
11          14,
12          878
13        ],
14        "id": 19995,
15        "original_language": "en",
16        "original_title": "Avatar",
17        "overview": "Un marine paraplégique, envoyé sur la lune Pandora pour une mission unique, est tiré en hélicoptère dans une zone dangereuse.",
18        "popularity": 126.67,
19        "poster_path": "/3npygfmEhqnmNTmDWhHLz1LPcbA.jpg",
20        "release_date": "2009-12-15",
21        "title": "Avatar",
22        "video": false,
23        "vote_average": 7.575,
24        "vote_count": 29955
25      },
26      {
27        "adult": false,
28        "backdrop_path": "/8rpDcsfLJypb06vREc0547VKqEv.jpg",
29        "genre_ids": [
30          878,
31          12,
32          28
33        ],
34        "id": 76600,
35        "original_language": "en",
36        "original_title": "Avatar: The Way of Water",
37        "overview": "Une dizaine d'années se sont écoulés depuis les précédents événements survenus sur Pandora. Jake Sully et Neytiri sont devenus parents. Leur vie idyllique, proche de la nature, est menacée lorsque la « Ressourcement Développement Administration », dangereuse organisation non-gouvernementale",
38        "popularity": 291.271,
39        "poster_path": "/hYeB9GpFaT7ysabBoGG5rbo9mF4.jpg",
40        "release_date": "2022-12-14",
41        "title": "Avatar : La Voie de l'eau",
42        "video": false,
43        "vote_average": 7.575,
44        "vote_count": 29955
45      }
46    ]
47   }
48 }

```

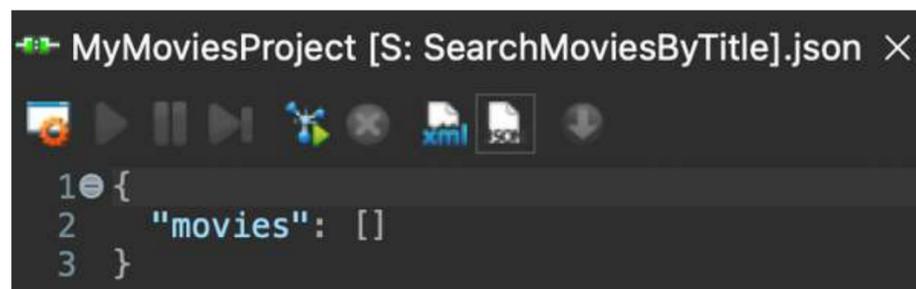


# 6.4 Using the IfExist step

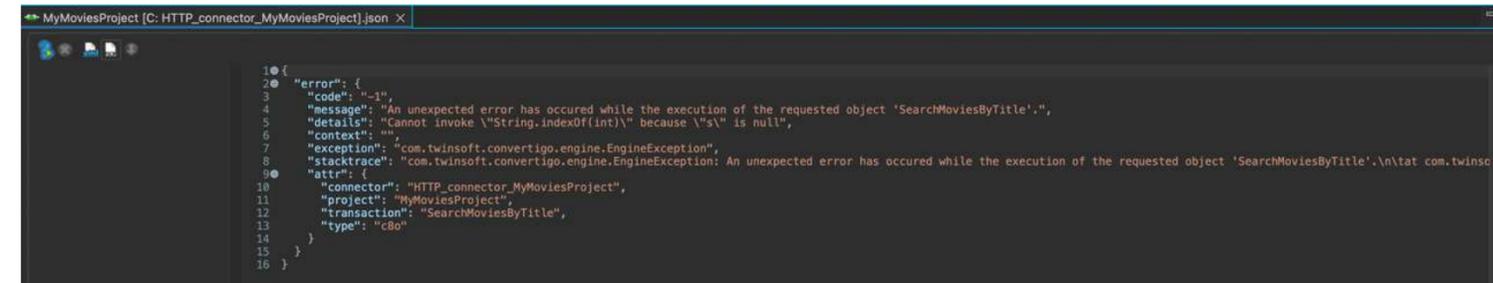
Let's run the error test case



Without a title,  
the sequence returns an empty array



The transaction returns an error,  
and an error tag is generated.



MyMoviesProject [C: HTTP\_connector\_MyMoviesProject].json



The IfExists step detects the error.

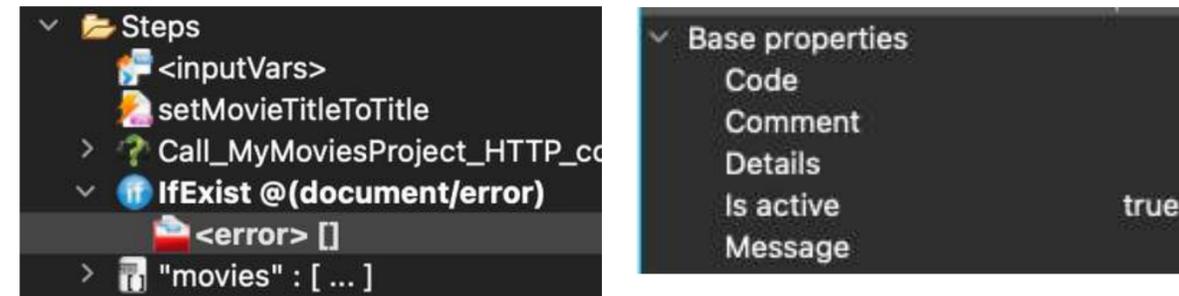
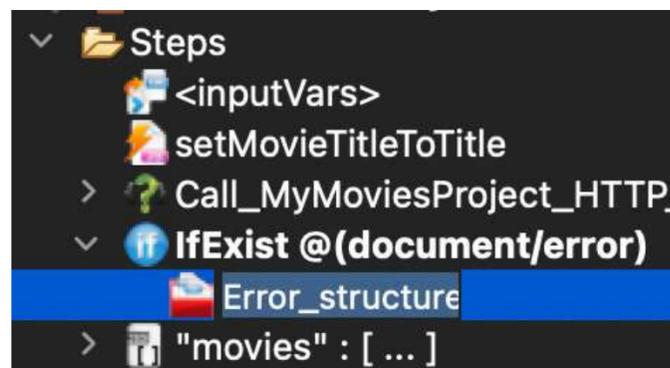
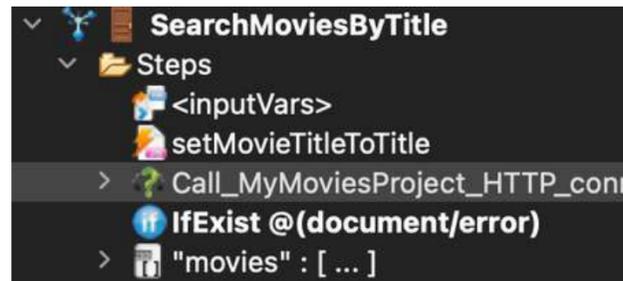
We are going to use it in combination to other steps to handle this error.



# 6.5 Using the Error Structure step

To report the error to the client, we use the **Error Structure step**,

Let's add an **Error structure step** in the **IfExist step**.

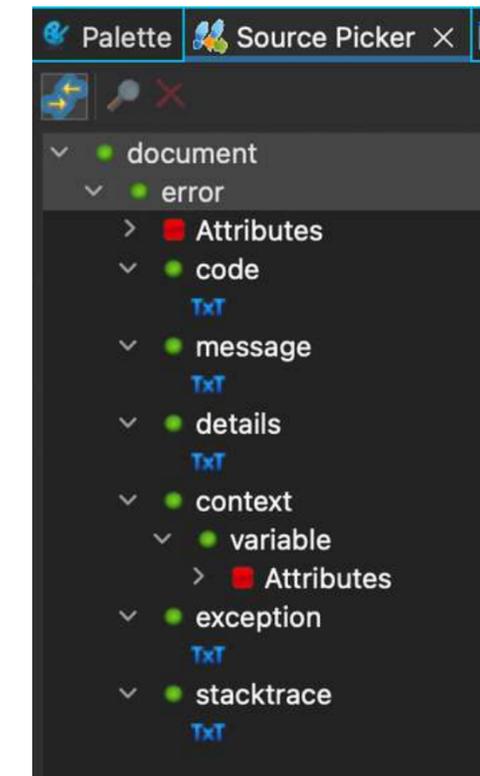


This step has the properties

- **Code** (error status code)
- **Message**
- **Details**

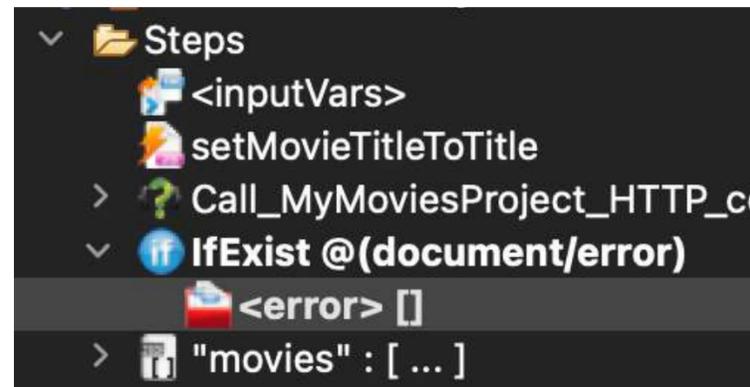
These properties can be **sourced from the original error message** returned by the API in the transaction.

In the source picker, the **XML Structure** of the **Error structure step** has the same properties.

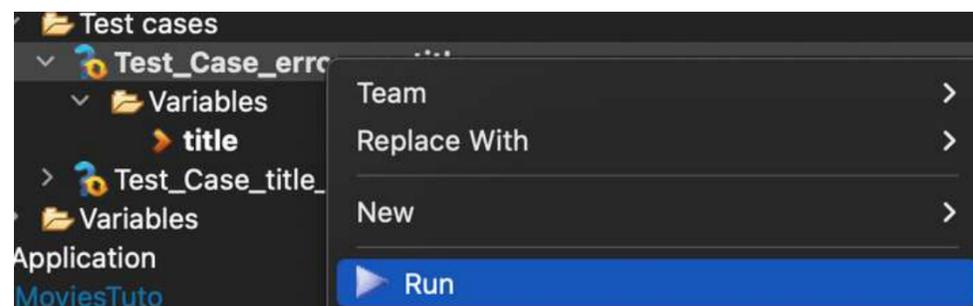


# 6.5 Using the Error Structure step

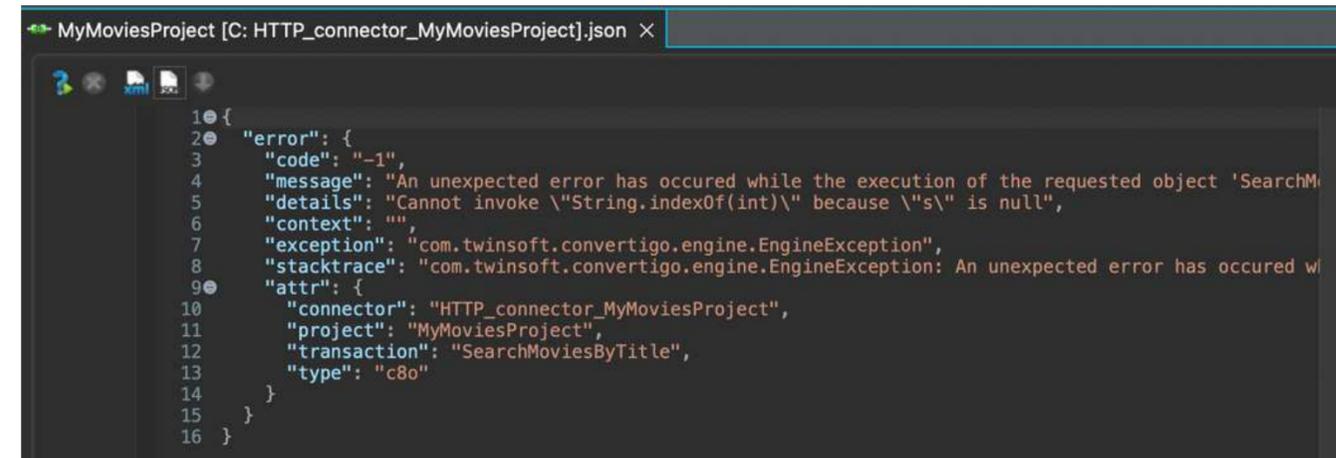
Let's see what difference it makes to have this Error structure step in the sequence.



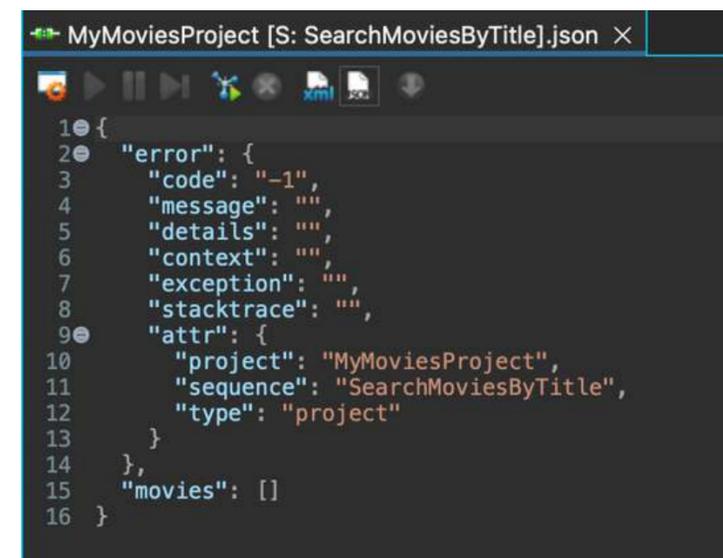
Let's run the error test case again.



The original error message returned by the transaction is the same



The sequence returns an empty array and an error.



# 6.5 Using the Error Structure step

By changing the properties of the Error structure step, we can customize the error returned by the sequence.

```

1 {
2   "error": {
3     "code": "-1",
4     "message": "",
5     "details": "",
6     "context": "",
7     "exception": "",
8     "stacktrace": "",
9     "attr": {
10    "project": "MyMoviesProject",
11    "sequence": "SearchMoviesByTitle",
12    "type": "project"
13  }
14 },
15 "movies": []
16 }
  
```



```

"error": {
  "code": "-1",
  "message": "",
  "details": ""
}
  
```



Let's customize our error message.

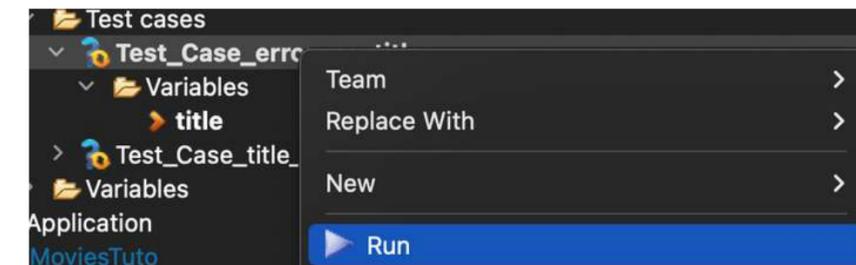
Base properties	
Code	
Comment	
Details	
Is active	true
Message	Some message

We can enter an error message directly in the message properties.

```

if IfExist @(document/error)
  <error> [] Some message
  "movies" : [ ... ]
  
```

The error message appears in the error structure step of the sequence.

Test cases

- Test\_Case\_err...
  - Variables >
  - title >
  - Test\_Case\_title\_ >
  - Variables >
- Application
- MoviesTuto

Run

When running the test case, the message will appear in the return of the execution of the sequence

```

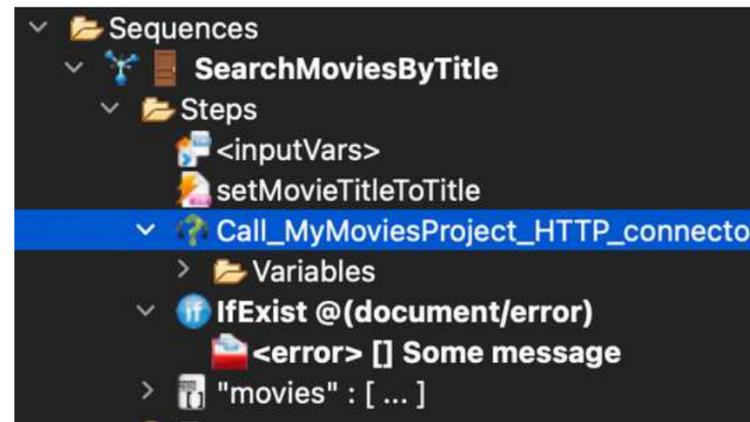
1 {
2   "error": {
3     "code": "-1",
4     "message": "Some message",
5     "details": "",
6     "context": "",
7     "exception": "",
8     "stacktrace": "",
9     "attr": {
10    "project": "MyMoviesProject",
11    "sequence": "SearchMoviesByTitle",
12    "type": "project"
13  }
14 },
15 "movies": []
16 }
  
```



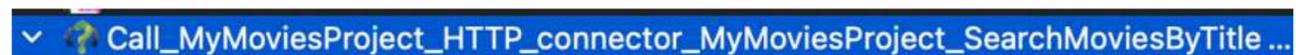
# 6.5 Using the Error Structure step

Now, let's **customize the error** returned by the sequence **dynamically**.

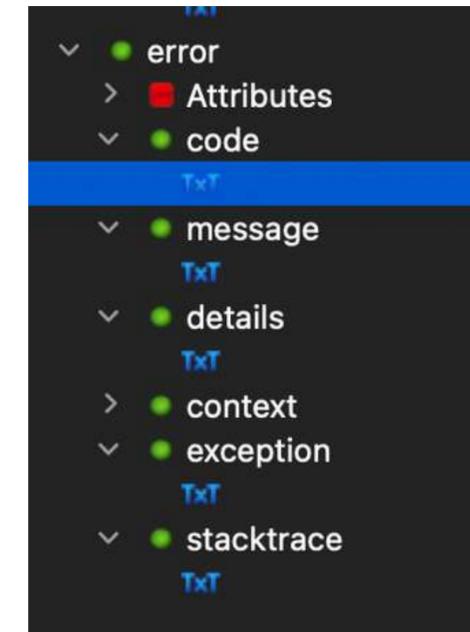
by using the original error message returned by the API, and the properties of the Error Structure step.



Click twice on the transaction call to display its XML structure in the source picker.



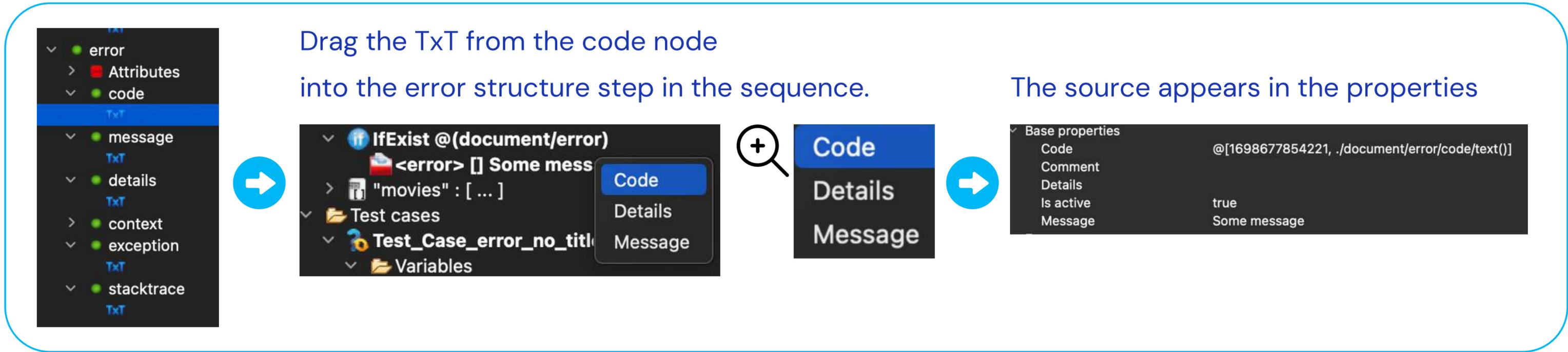
In the source picker, we can see the **nodes code, message and details**. We are going to use these nodes to **source the properties** of the Error Structure step.



# 6.5 Using the Error Structure step

Drag the TxT from the code node into the error structure step in the sequence.

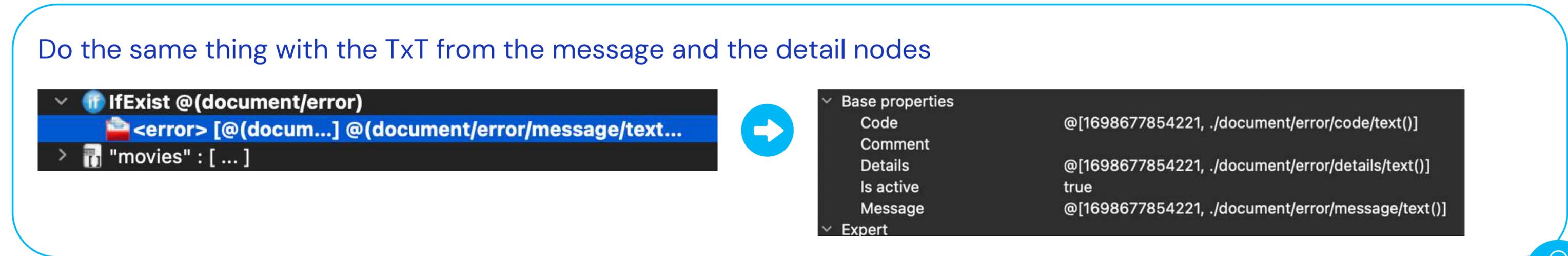
The source appears in the properties



The diagram illustrates the process of configuring an error structure step. On the left, a tree view shows the 'error' node expanded, with 'code' selected. An arrow points to a sequence step 'IfExist @(document/error)' which contains an error structure. A context menu is open over the 'code' node, with 'Code' selected. A magnifying glass highlights the 'Code' property in the properties panel, which is set to '@[1698677854221, ./document/error/code/text()]'. An arrow points to the final properties panel where the 'Code' property is visible.



Do the same thing with the TxT from the message and the detail nodes

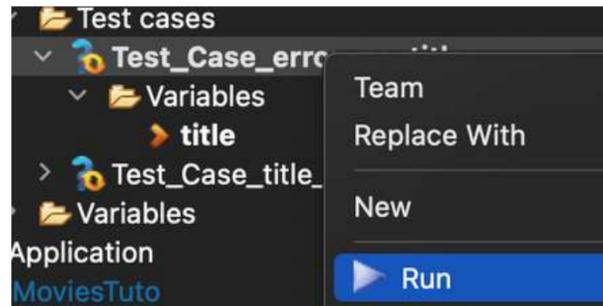


The diagram illustrates the process of configuring an error structure step with multiple TxT nodes. On the left, a sequence step 'IfExist @(document/error)' is shown with an error structure containing 'message' and 'details' nodes. An arrow points to the properties panel where the 'Code', 'Details', and 'Message' properties are visible, each with its corresponding source path.



# 6.5 Using the Error Structure step

When we run the test case, the error returned by the sequence has the same code, message and details as the original error message returned by the API.



Error returned by the sequence

Original error message returned by the API

```

1 {
2   "error": {
3     "code": "-1",
4     "message": "An unexpected error has occurred while the execution of the requested object 'SearchM
5     "details": "Cannot invoke 'String.indexOf(int)' because 's' is null",
6     "context": "",
7     "exception": "",
8     "stacktrace": "",
9     "attr": {
10      "project": "MyMoviesProject",
11      "sequence": "SearchMoviesByTitle",
12      "type": "project"
13    }
14  },
15  "movies": []
16 }

```

```

1 {
2   "error": {
3     "code": "-1",
4     "message": "An unexpected error has occurred while the execution of the requested object 'SearchM
5     "details": "Cannot invoke 'String.indexOf(int)' because 's' is null",
6     "context": "",
7     "exception": "com.twinsoft.convertigo.engine.EngineException",
8     "stacktrace": "com.twinsoft.convertigo.engine.EngineException: An unexpected error has occurred w
9     "attr": {
10      "connector": "HTTP_connector_MyMoviesProject",
11      "project": "MyMoviesProject",
12      "transaction": "SearchMoviesByTitle",
13      "type": "c80"
14    }
15  }
16 }

```



# 6.6 Using the Return step

```

1 {
2   "error": {
3     "code": "-1",
4     "message": "An unexpected error has occurred while the execution of the requested object 'SearchM
5     "details": "Cannot invoke 'String.indexOf(int)' because 's' is null",
6     "context": "",
7     "exception": "",
8     "stacktrace": "",
9     "attr": {
10      "project": "MyMoviesProject",
11      "sequence": "SearchMoviesByTitle",
12      "type": "project"
13    }
14  },
15  "movies": []
16 }

```

After the error structure step, the sequence didn't stop and the following steps were executed. That's why we see an empty movies array after the error.



To stop the sequence after an error,

```

v IfExist @(document/error)
  <error> [@(docum...) @(document/error/message
> "movies" : [ ... ]

```



we add a **Return step** after the **Error Structure step** in the **IfExist step**.

```

v IfExist @(document/error)
  <error> [@(docum...) @(document/error/message/t
  return
> "movies" : [ ... ]

```

The sequence execution is stopped and the empty movies array disappears from the result of the sequence.



```

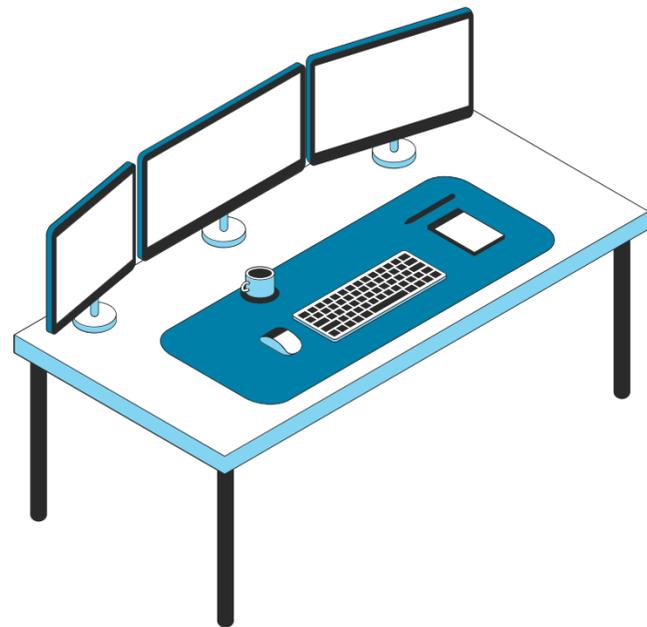
1 {
2   "error": {
3     "code": "-1",
4     "message": "An unexpected error has occurred while the execution of
5     "details": "Cannot invoke 'String.indexOf(int)' because 's' is
6     "context": "",
7     "exception": "",
8     "stacktrace": "",
9     "attr": {
10      "project": "MyMoviesProject",
11      "sequence": "SearchMoviesByTitle",
12      "type": "project"
13    }
14  }
15 }

```



# 7 – Collaboration with Git

How to share your projects  
with Git Versioning.



**7.1** Git basics with Convertigo

---

**7.2** Git Repositories View

---

**7.3** Git Staging View

---

**7.4** Compare mode

---

**7.5** Commit your changes

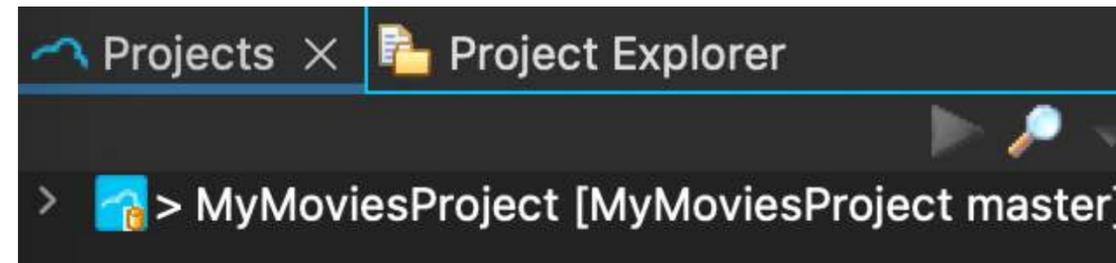
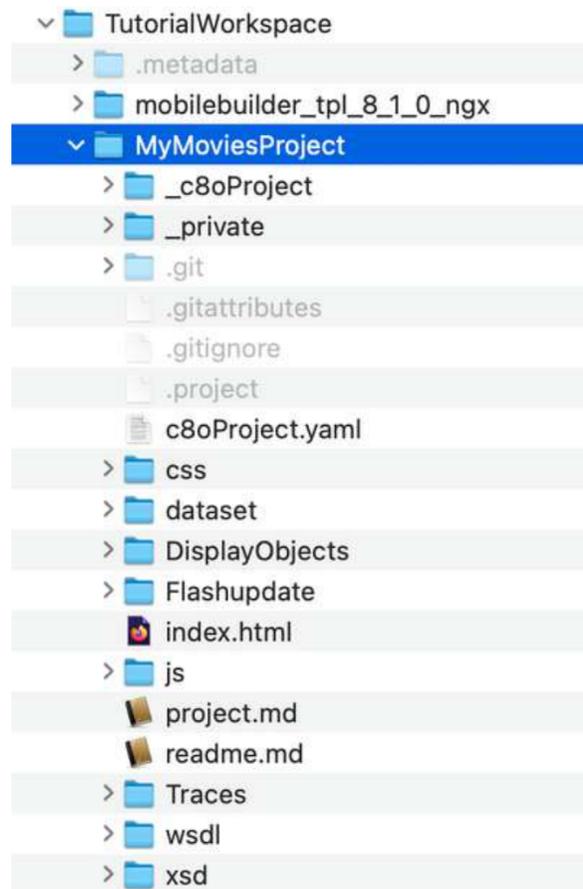
---

**7.6** Clone a project

# 7.1 Git basics with Convertigo

When you create a new project, a **Git Repository** is automatically created.

In the Projects folder, the name of your project is followed by the **name of the branch** you're currently working on.

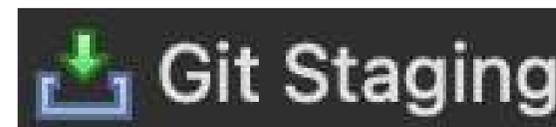


In the studio interface, two views are used to manage Git in your projects.

- Git Repositories

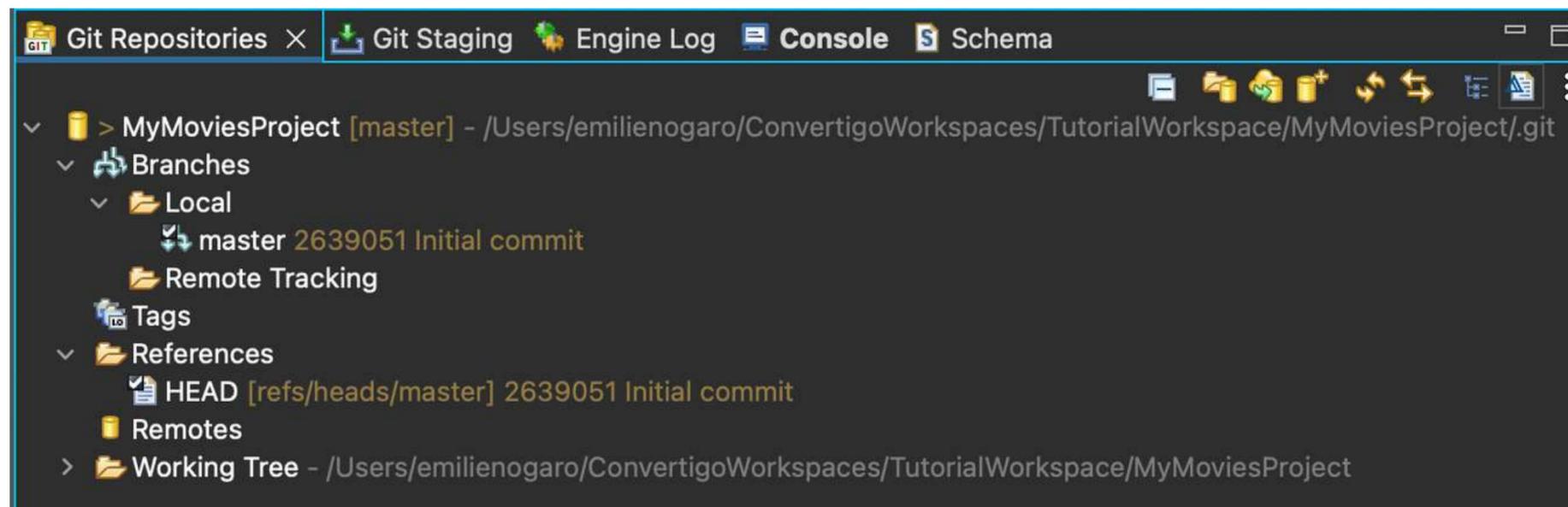


- Git Staging

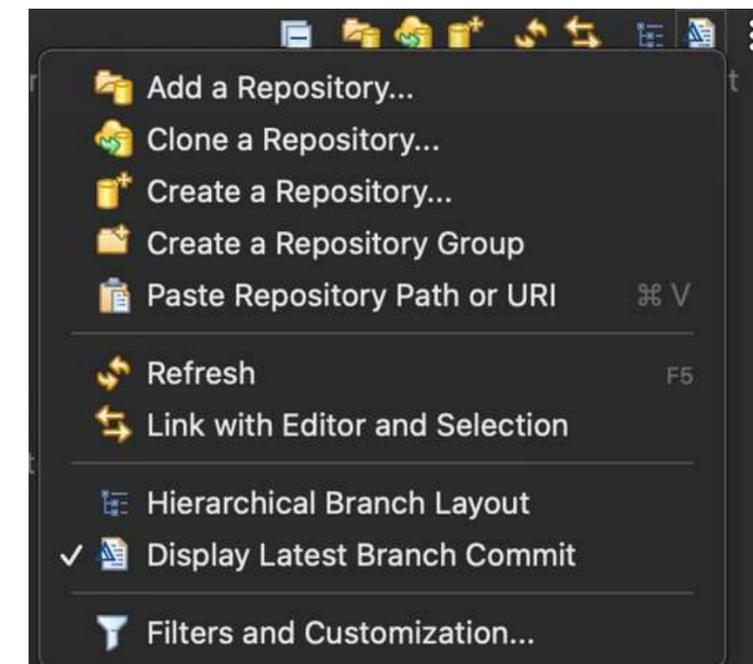


# 7.2 Git Repositories View

In the **Git Repositories View**, you can see the Git Repositories of **all the projects in your workspace**.



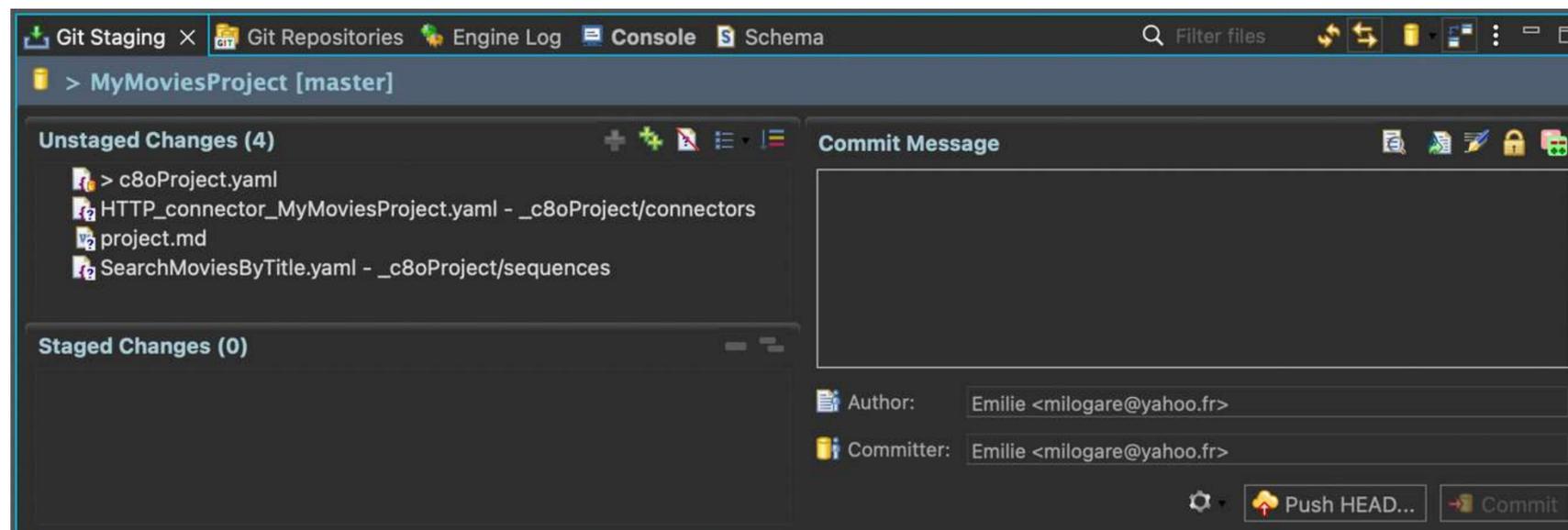
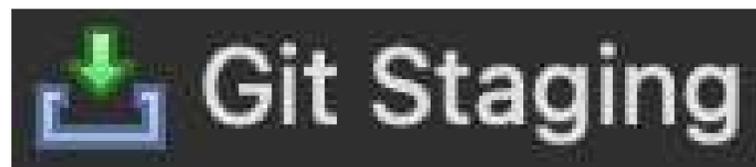
It provides Eclipse features to manage your Git repositories.



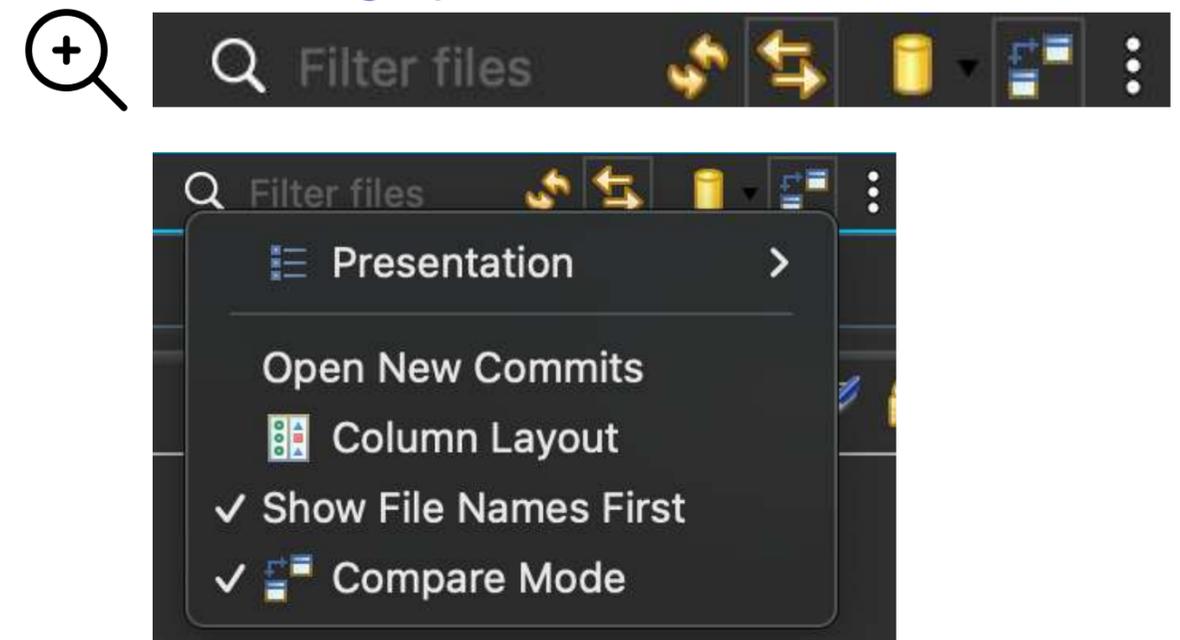
# 7.3 Git Staging View

In the **Git Staging view**, you can manage your **git workflow**, and **commit your changes** to your **local and remote directories**.

The files that have been modified since the last commit are shown in **Unstaged changes**.

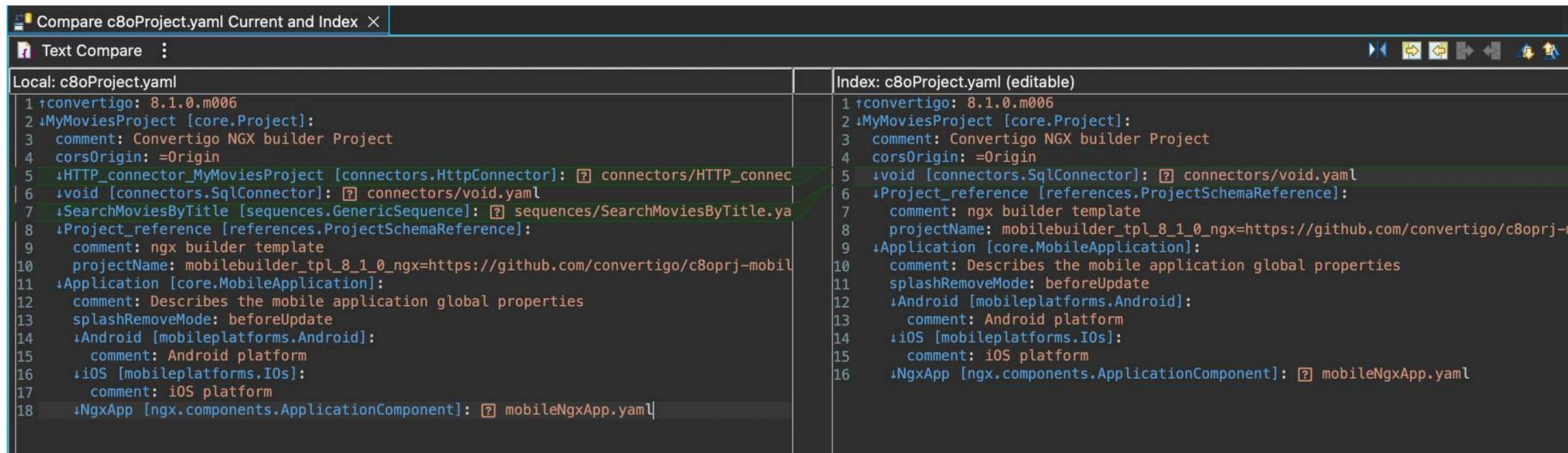


It provides Eclipse features to manage your Git workflow.



# 7.4 Compare Mode

With the **Compare Mode**, you can display the **differences with the previous commit**, and **resolve conflicts** when necessary.



The screenshot shows a 'Text Compare' window with two panels. The left panel is titled 'Local: c8oProject.yaml' and the right panel is 'Index: c8oProject.yaml (editable)'. Both panels show a list of 18 lines of YAML configuration. The 'Local' version includes an 'HTTP\_connector' and a 'SearchMoviesByTitle' sequence, while the 'Index' version includes a 'void' connector and a 'Project\_reference'.

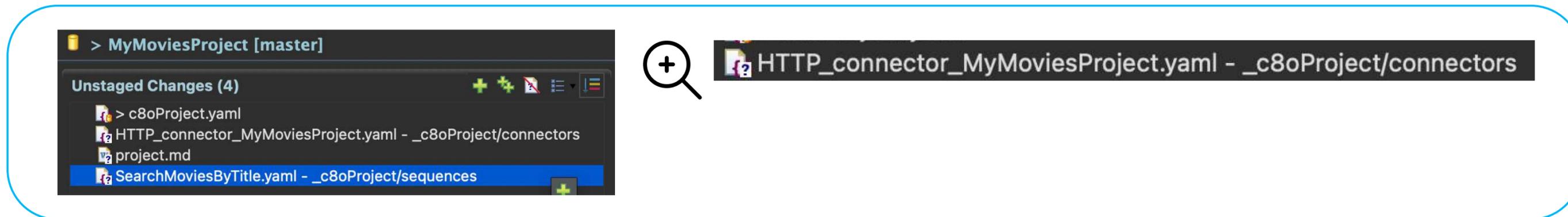


These icons represent **features** allowing you to **manipulate the files** : navigate to the next or previous changes, swap the views, copy changes from one view to the other...



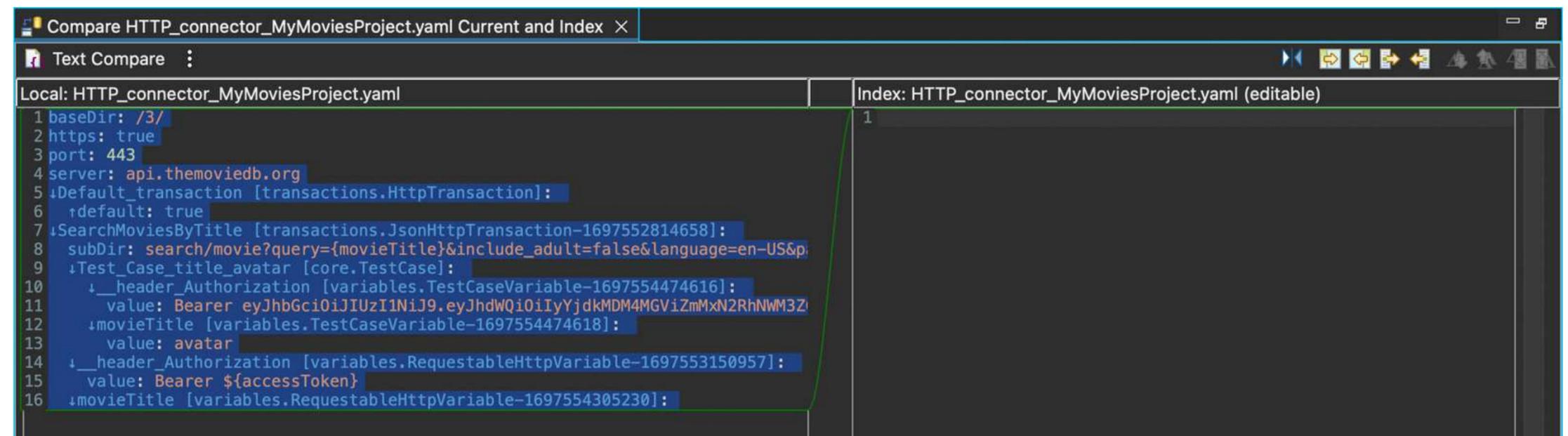
# 7.4 Compare Mode

Right-click twice on a file in the Staging view to open the compare mode.



Compare HTTP\_connector\_MyMoviesProject.yaml Current and Index X

You can see the changes since the last commit :  
Here the **index view** is empty because the connector was created after the last commit



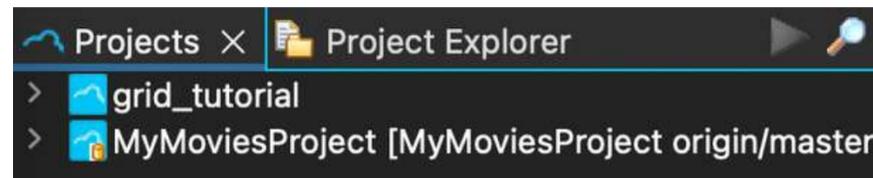
# 7.5 Create a repository

When you **create a new project** in your workspace, a **Git Repository is automatically created**.

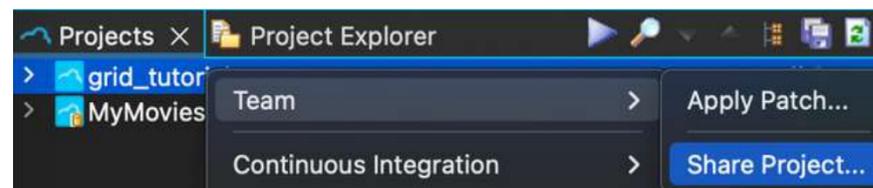
But if you **import a project from a .car file**, you have to **create it manually**.

Let's say we want to create a Git Repository for the the project grid\_tutorial

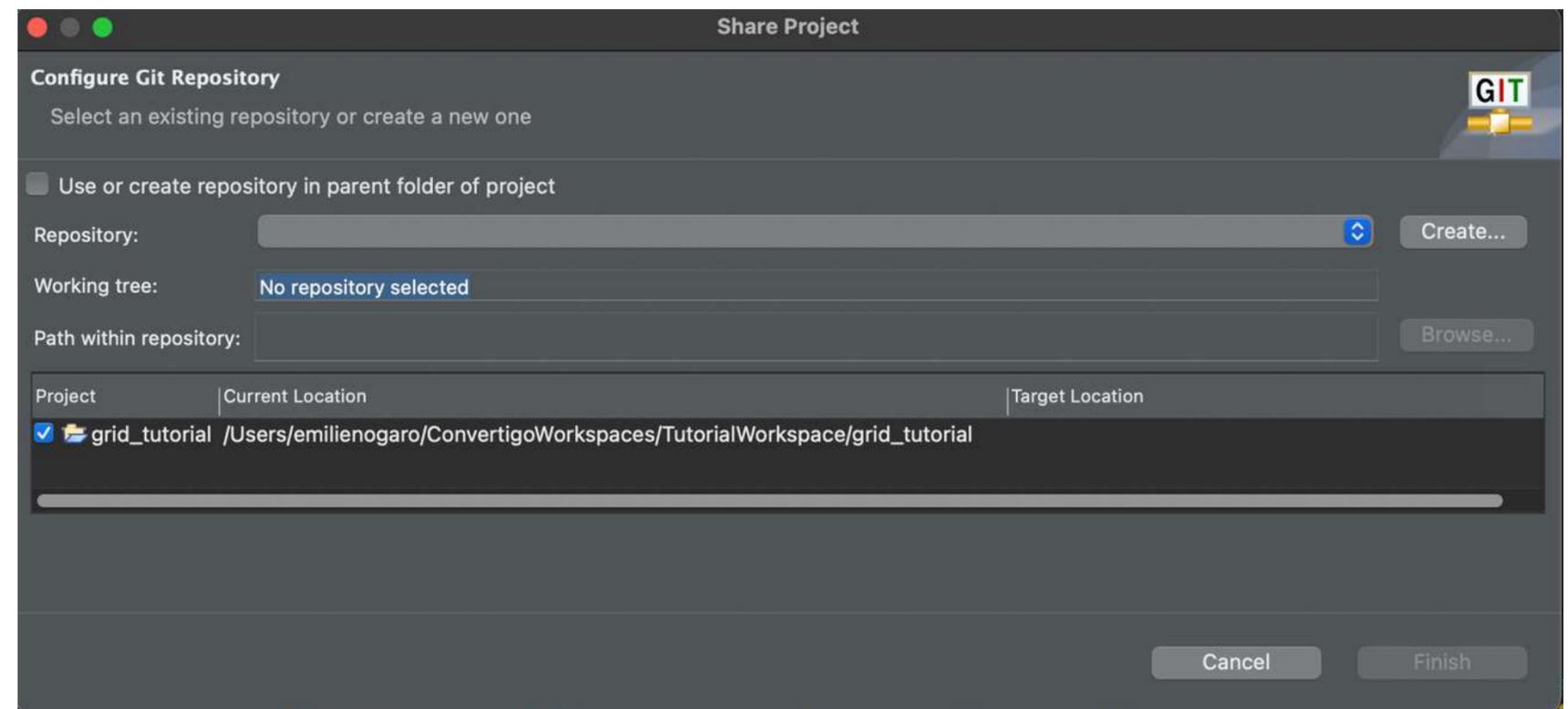
Right-click on the project grid\_tutorial in the Projects view.



Select Team > then Share Project

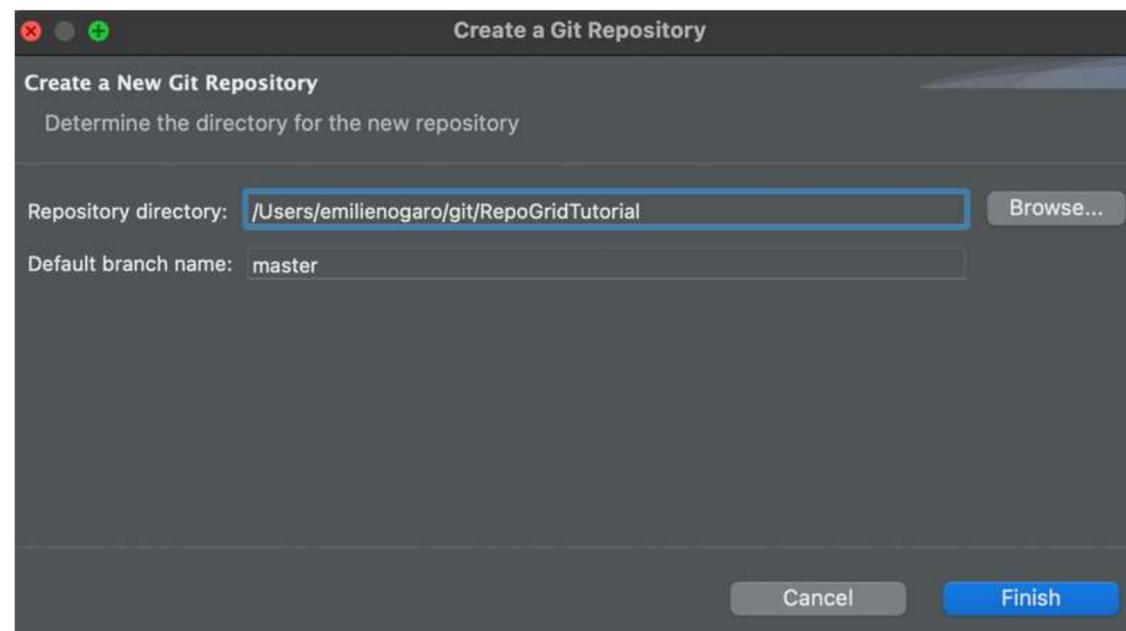


In the Configure Git Repository window of the Share Project Window, click on Create to create the repository

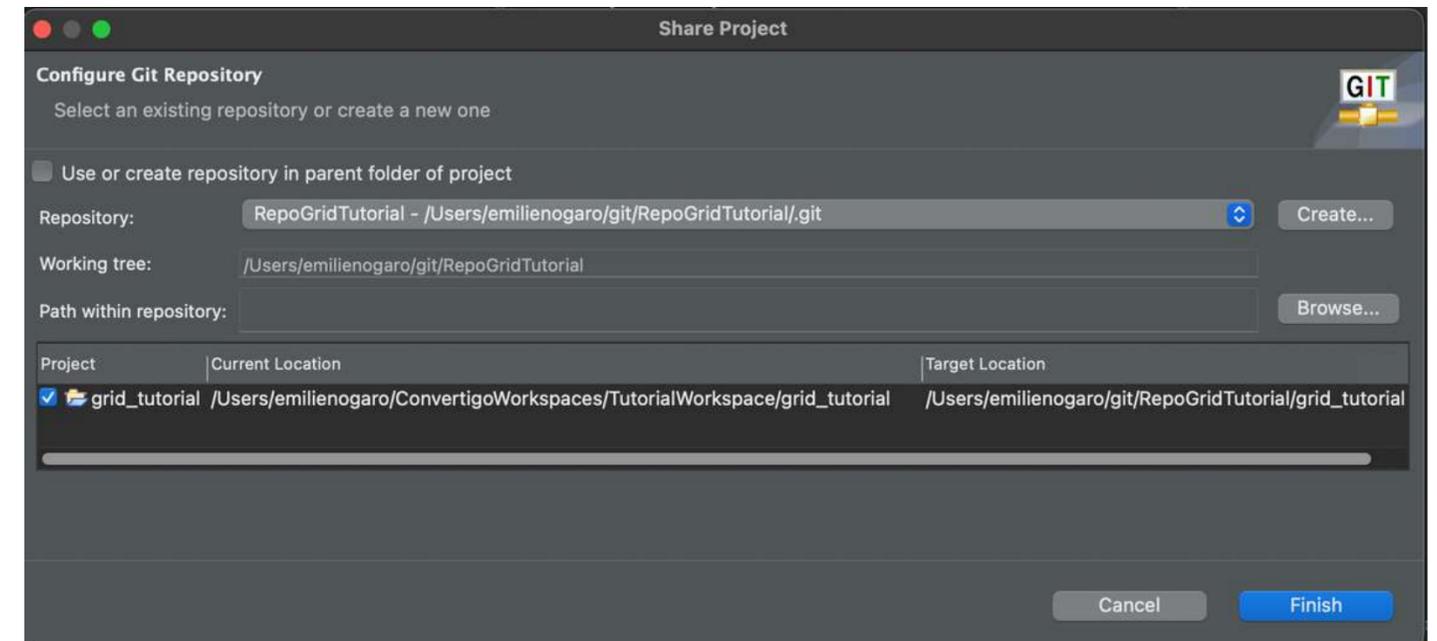


# 7.5 Create a repository

In the **Create a new Git Repository** window, change the repository name (repository by default) to RepoGridTutorial. Then click on **Finish**.

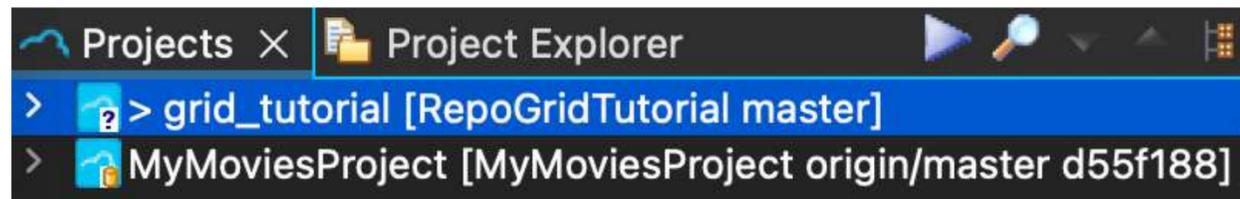


In the **Configure Git Repository** window, you can see the repository name and its path. Click on **Finish**.

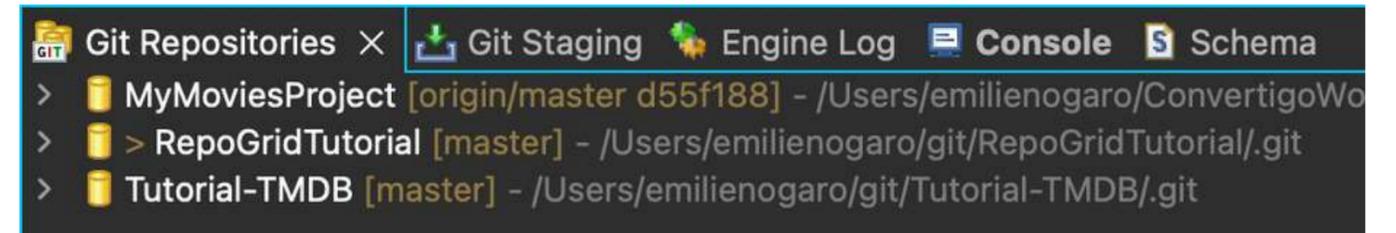


## 7.5 Create a repository

In the **Projects view**,  
the **repository name** and the **branch name**  
appears after the project name.



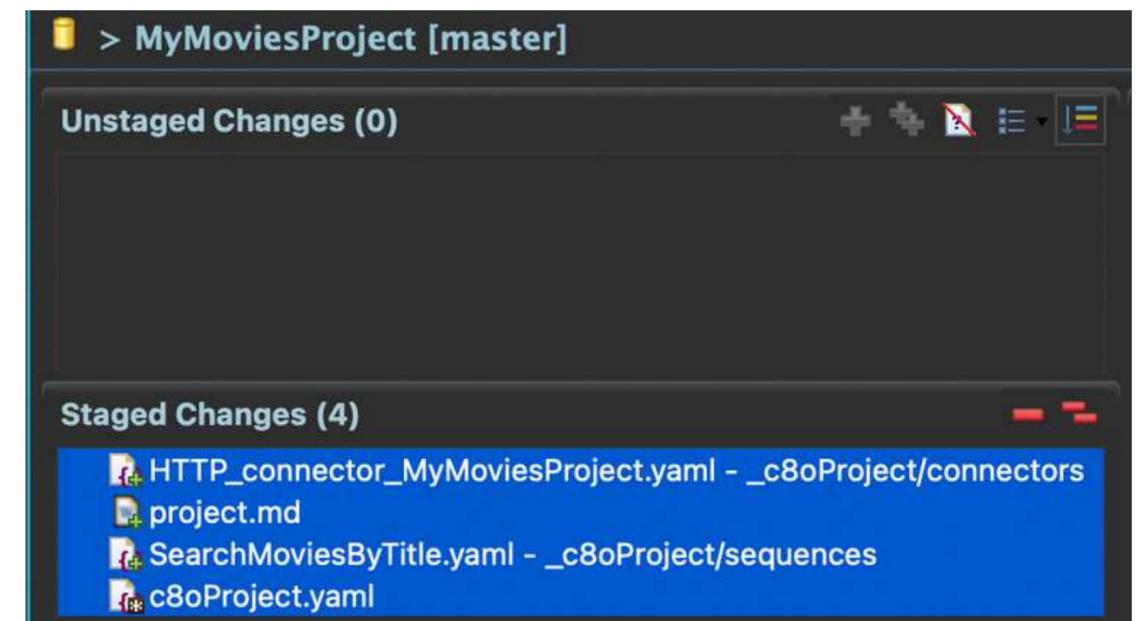
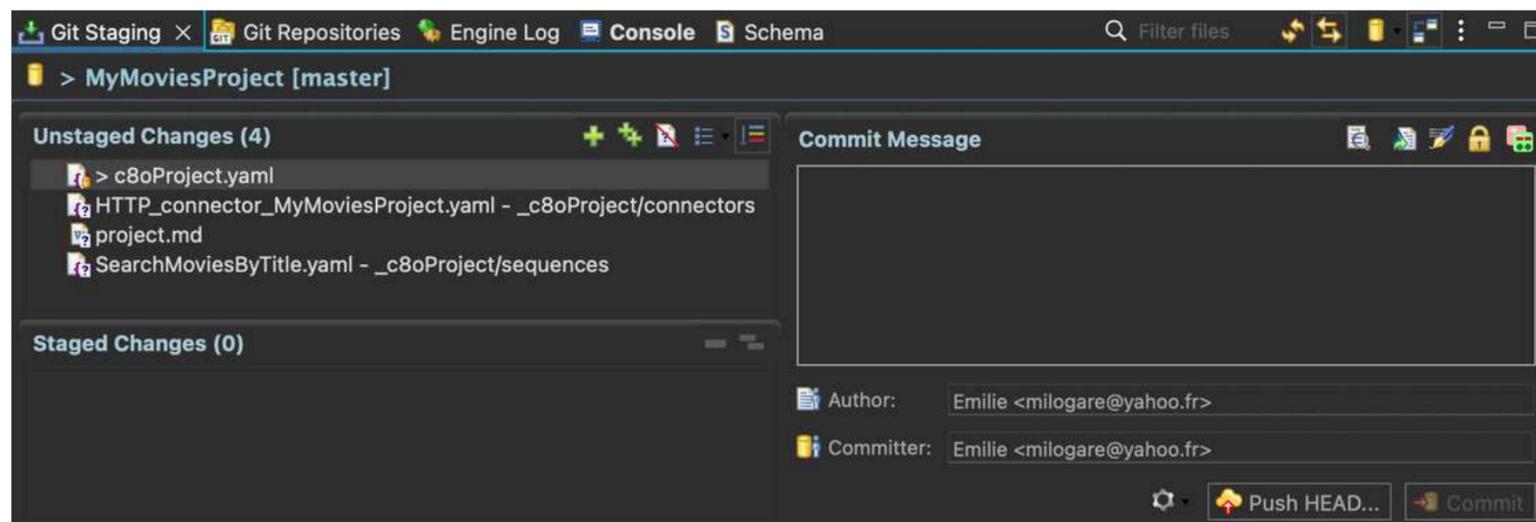
In the **Git Repositories view**,  
the **repository name**, the **branch name**  
and the **path to the Git repository** appears.



# 7.6 Commit your changes

Let's say you have **made a few changes** in your project and you want to **commit them on a Git repository**.

Go the Git Staging view and stage your files.



Stage your files with the green cross



One by One



Or All at Once

You can also Unstage them with the red line



One by One

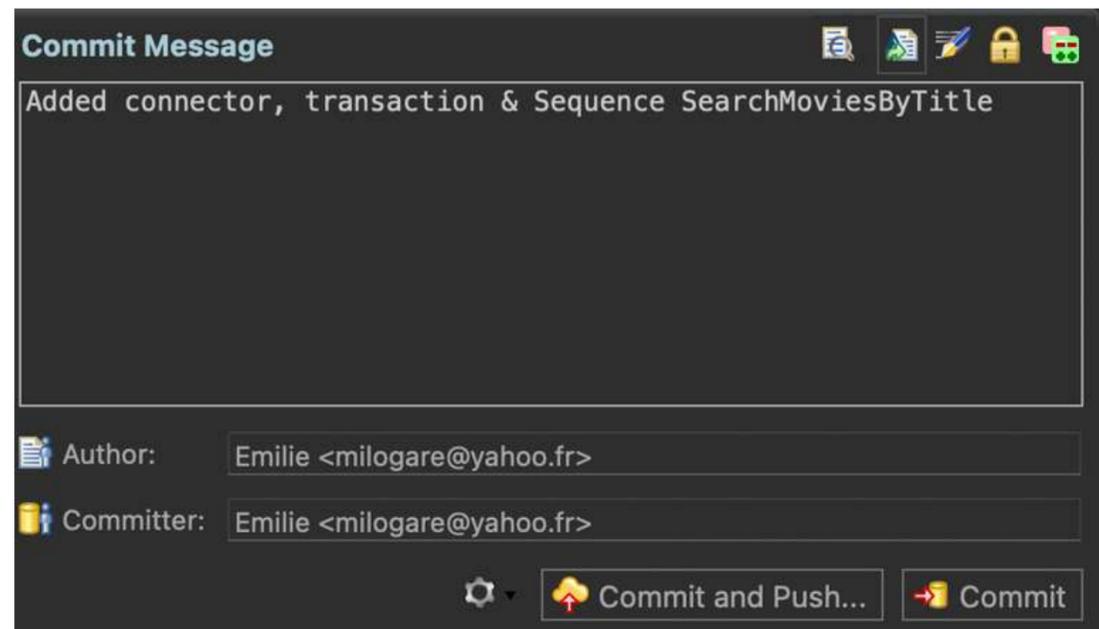


Or All at Once

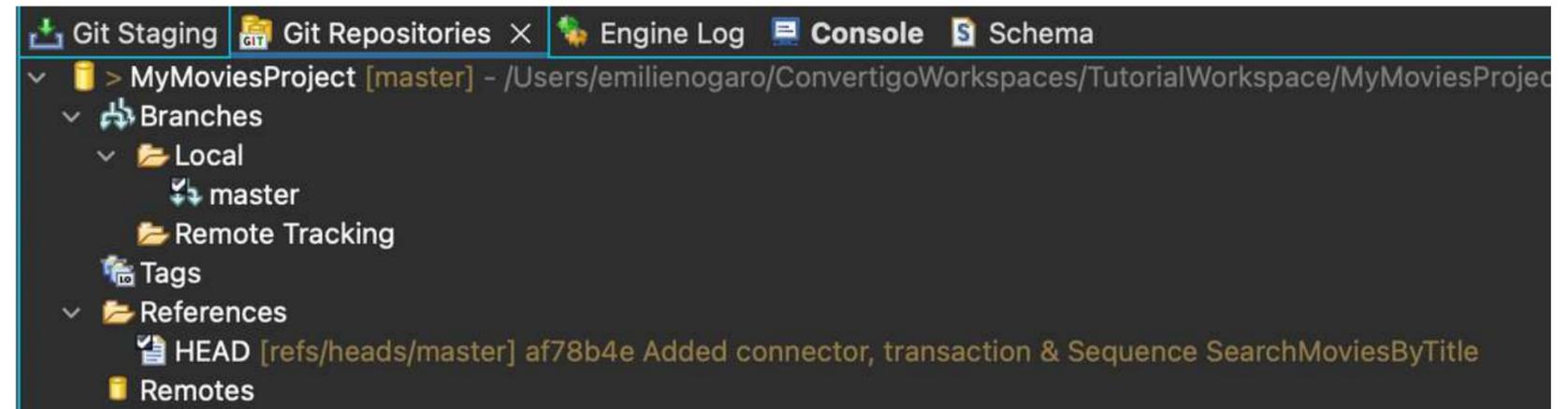


# 7.6 Commit your changes

Add a commit message and click on **Commit**.



Your changes have been **committed to your local Git repository**. In the **Git Repositories** view, you can see the **latest commit** in the **References** folder.



# 7.6 Commit your changes

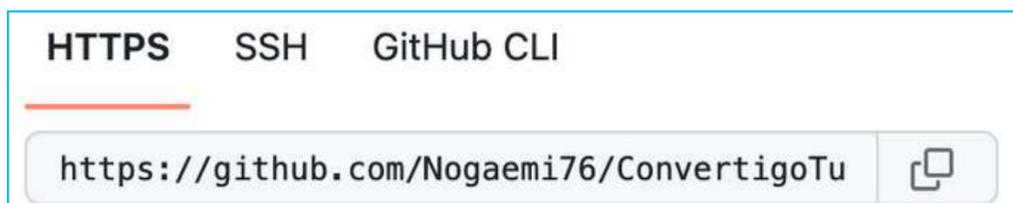
At this stage, only your **local Git repository** has been initialized.

Let's add a **remote repository** to your project.

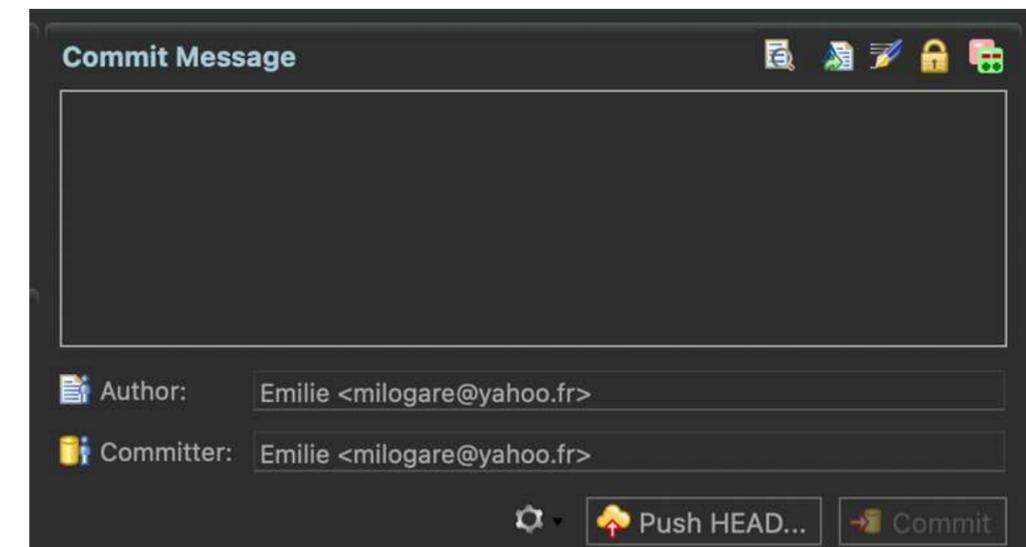
Create an **empty remote repository** in GitHub or GitLab.



Copy your **repo URI** to the clipboard.



In the **Git Staging view**

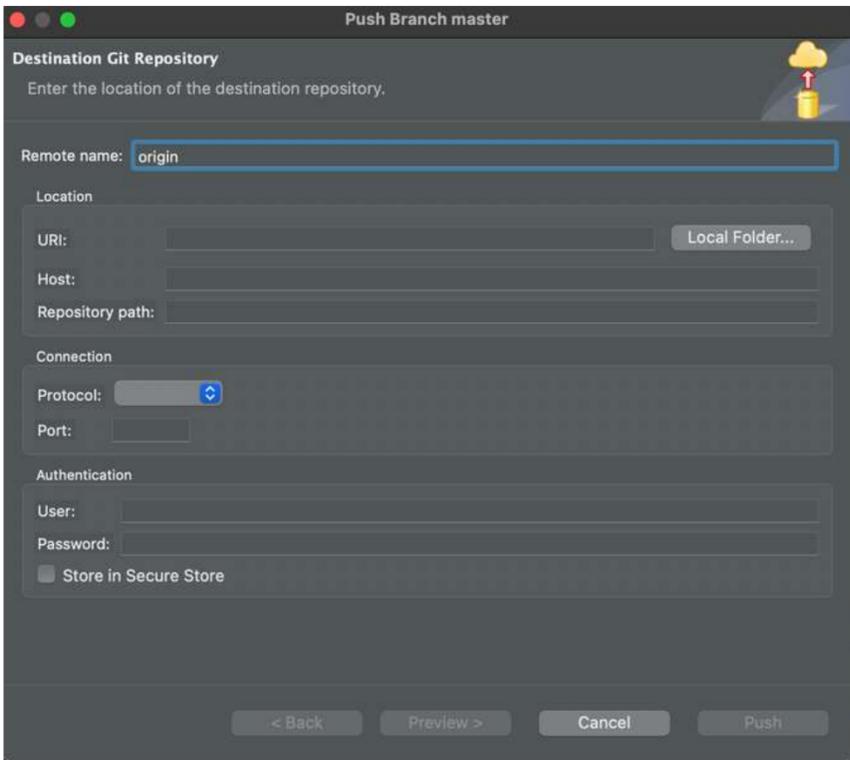


Click on **Push HEAD.**



# 7.6 Commit your changes

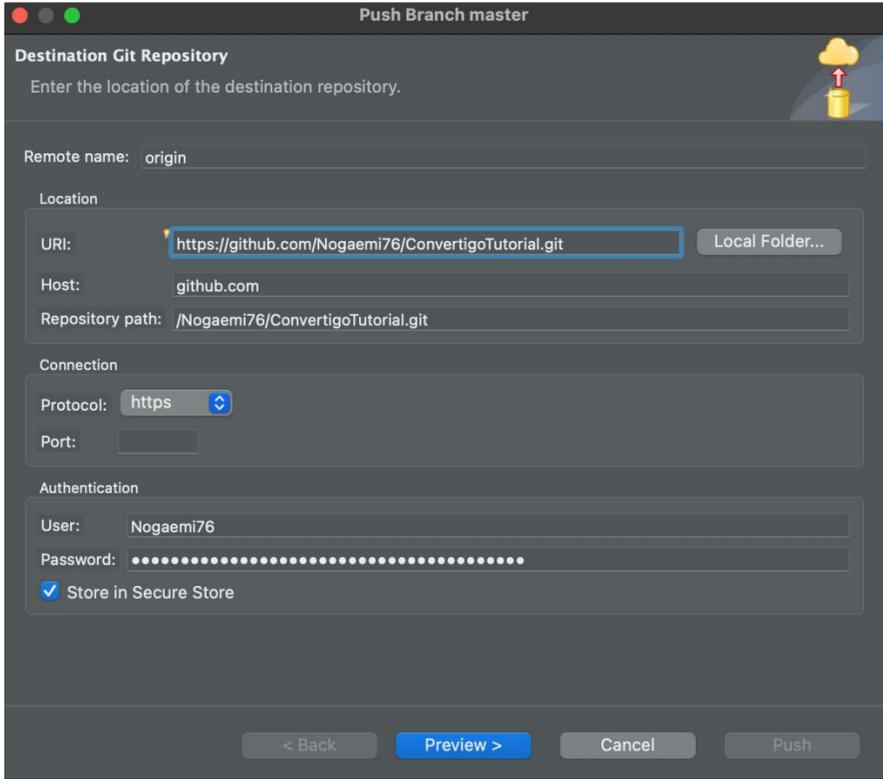
In the **Destination Git Repository** window of the **Push Branch Master** window, paste the **URI** in the **URI** field.



The screenshot shows the 'Push Branch master' window. The 'Destination Git Repository' section is active, with the 'Remote name' field set to 'origin'. The 'URI' field is empty. Below the screenshot, a blue arrow points down to a dark grey box containing the text 'URI: https://github.com/Nogaemi76/ConvertigoTutorial.git'.



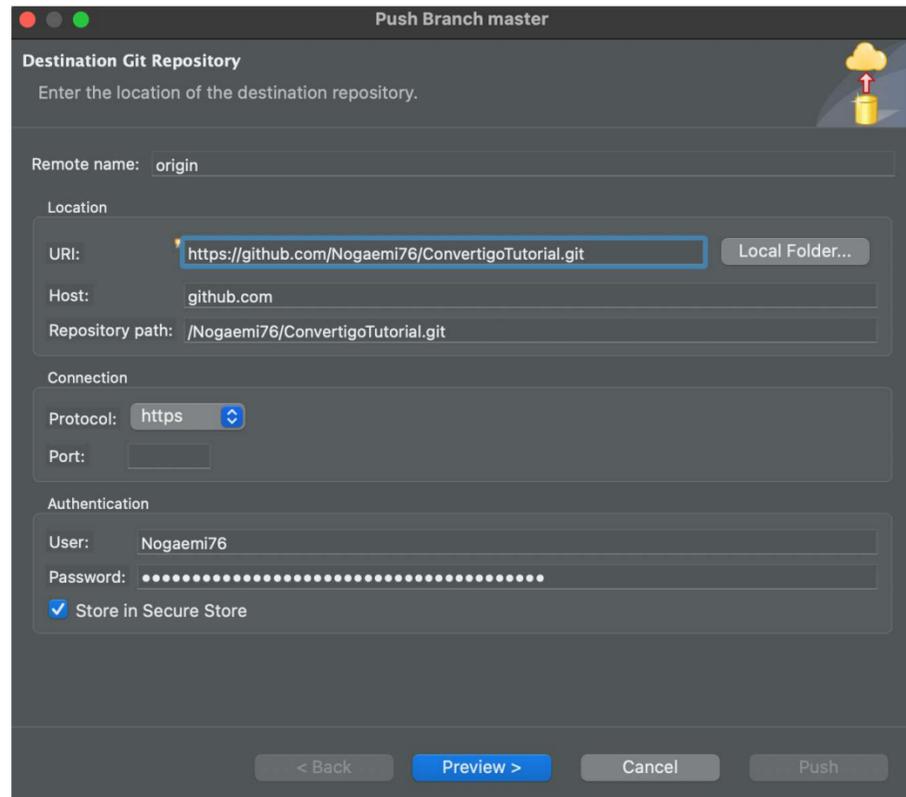
The other fields will update automatically



The screenshot shows the 'Push Branch master' window. The 'Destination Git Repository' section is active, with the 'Remote name' field set to 'origin'. The 'URI' field is populated with 'https://github.com/Nogaemi76/ConvertigoTutorial.git'. The 'Host' field is 'github.com' and the 'Repository path' is '/Nogaemi76/ConvertigoTutorial.git'. The 'Protocol' is 'https'. The 'User' field is 'Nogaemi76' and the 'Password' field is filled with dots. The 'Store in Secure Store' checkbox is checked. Below the screenshot, a blue arrow points down to a dark grey box containing the text 'Preview >' and the text 'Click on Preview >'.

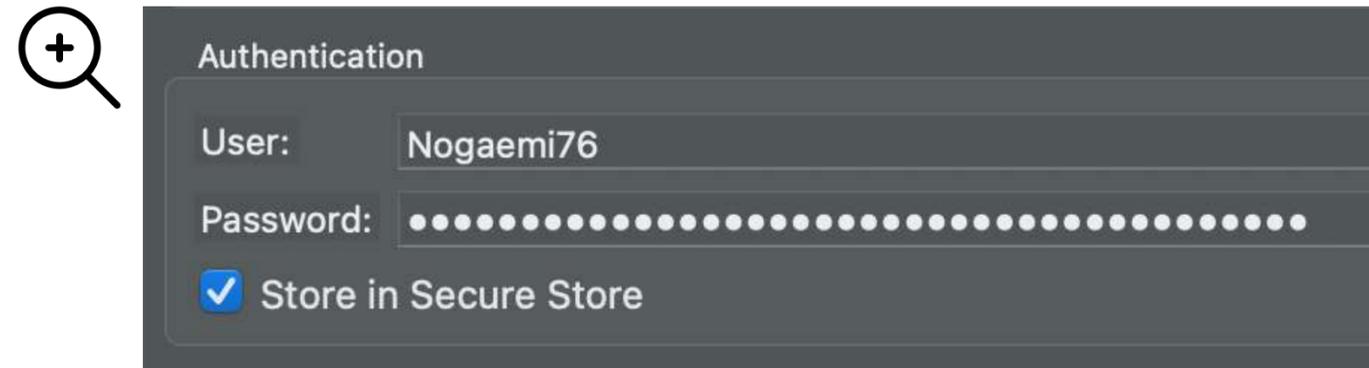


# 7.6 Commit your changes



Reminder : In the Authentication part of the Push Branch Master window.

- User is your GitHub Username
- Password is a Personal access token from GitHub



 Settings / Developer Settings



 Personal access tokens

Fine-grained tokens

Beta

Tokens (classic)



Personal access tokens (classic)

Generate new token

Revoke all



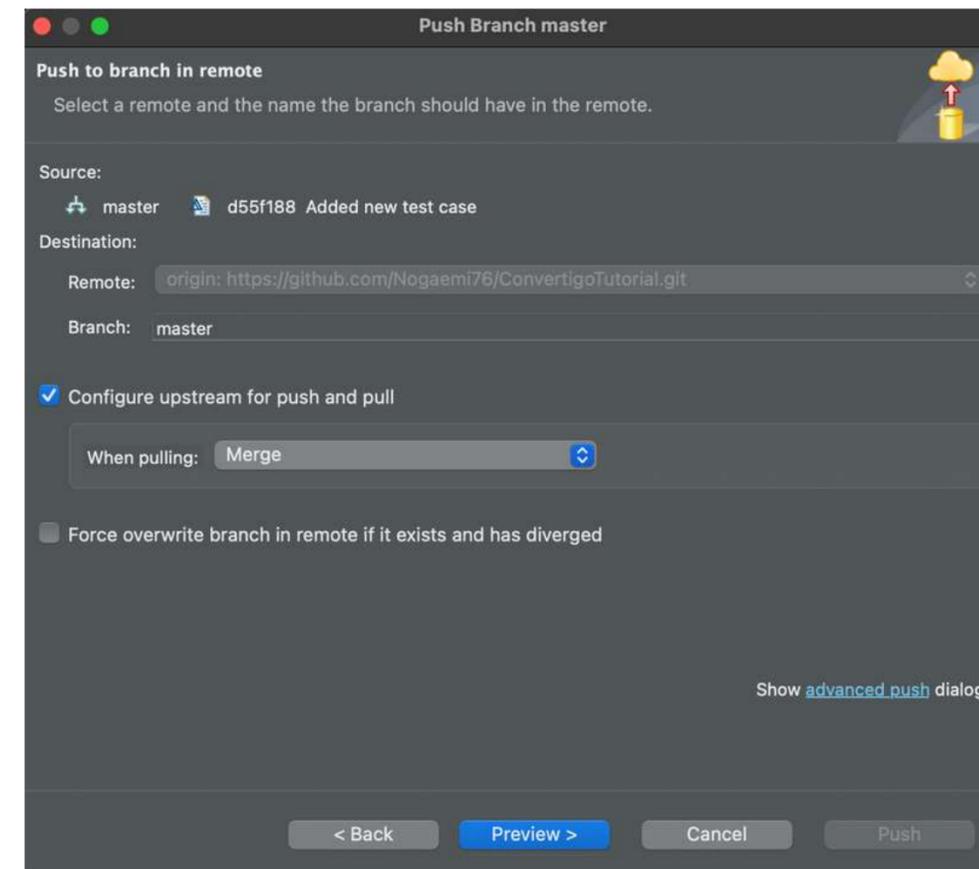
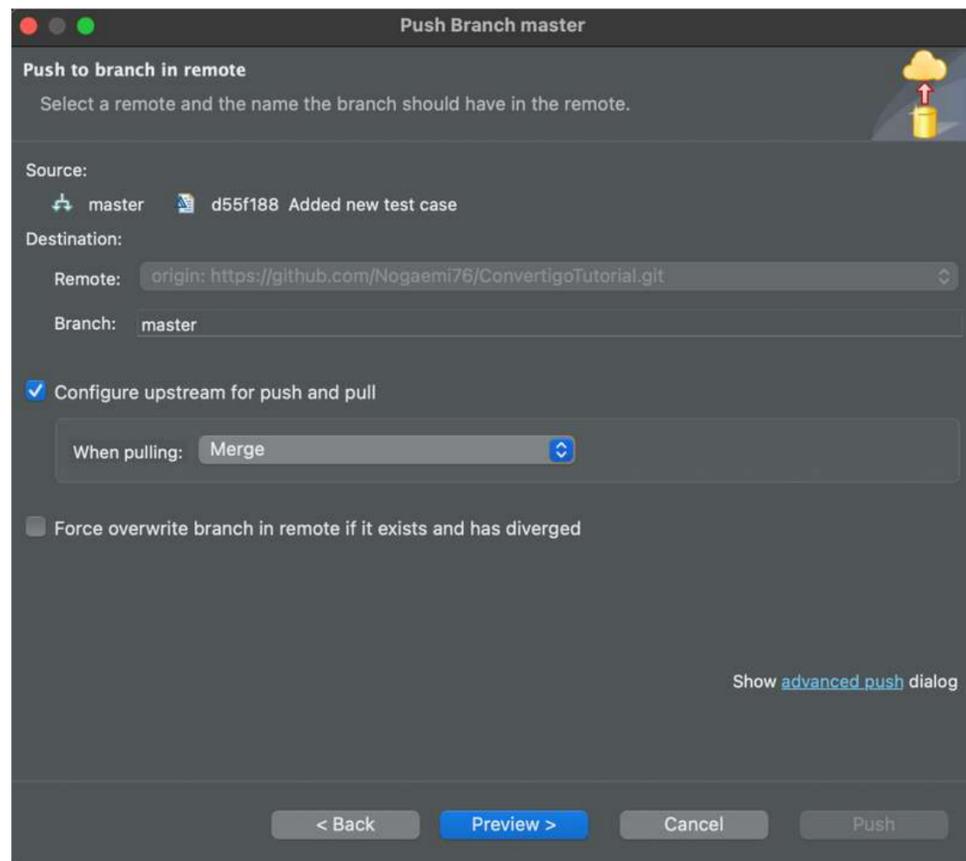
# 7.6 Commit your changes

The **Push to branch in remote** window appears.  
You can change the remote branch if necessary.

Click on **Preview >**.

The **Push to branch in remote** window appears.

Click on **Preview >**

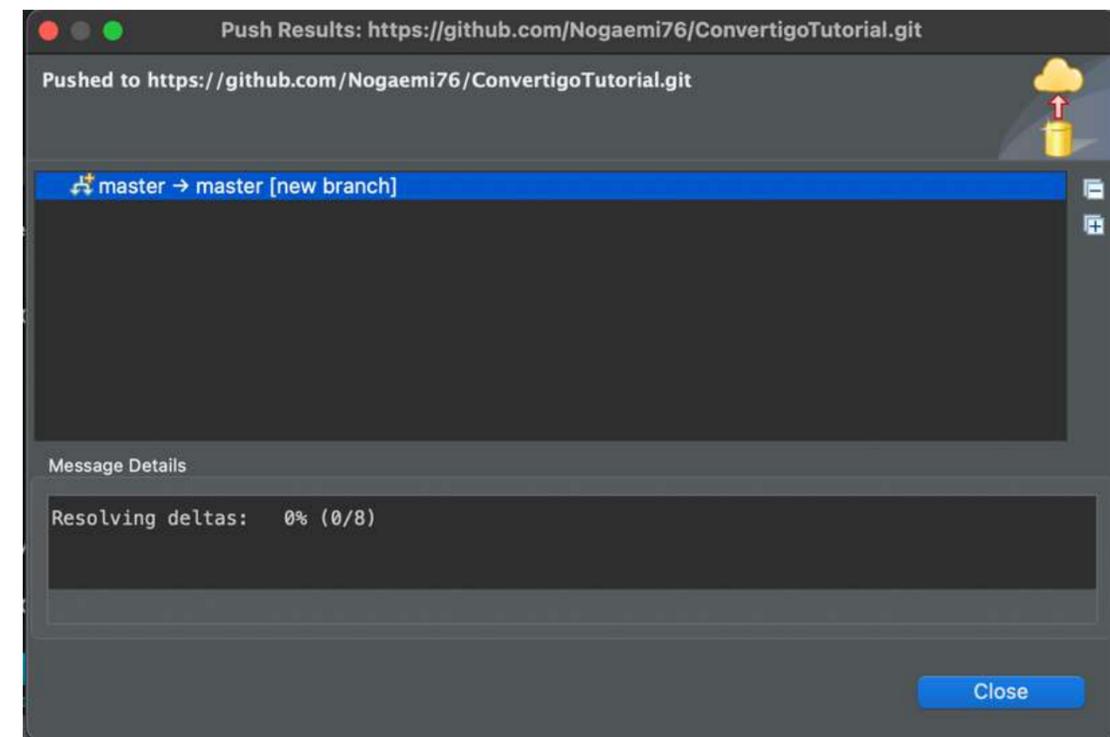
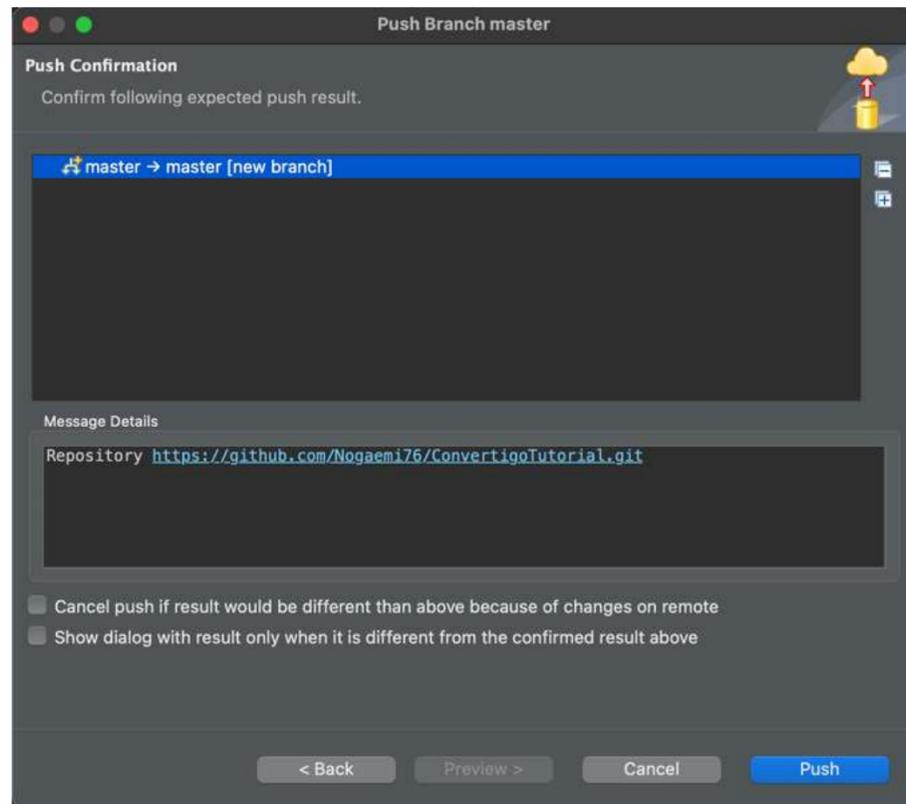


# 7.6 Commit your changes

The **Push Confirmation** window appears.

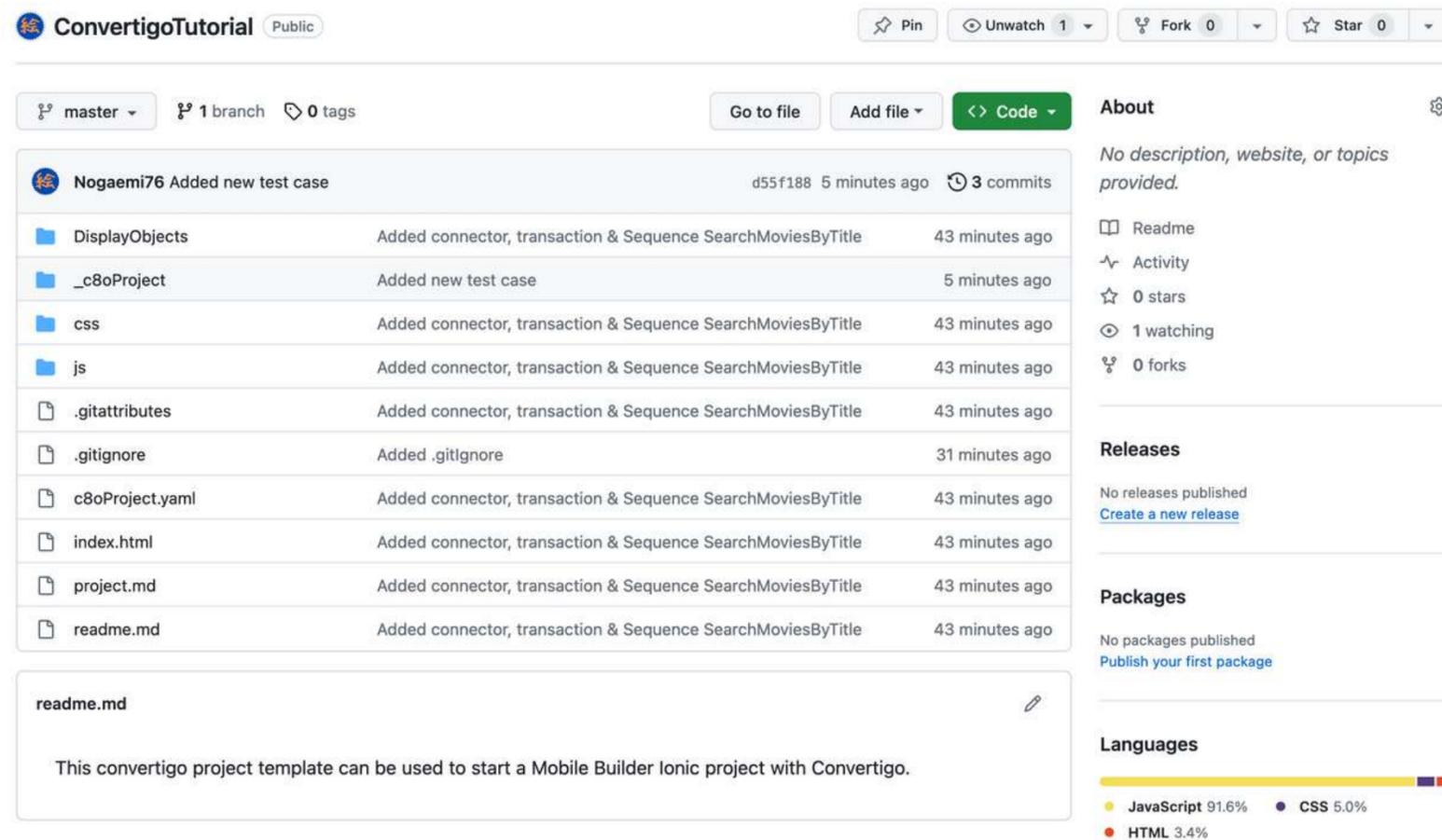
Click on **Push**  
to **push your project on your remote repository.**

A **Push Results** window appears  
to confirm that your project has been pushed  
on your remote repository.



# 7.6 Commit your changes

Your project appears in your **remote repository**.



**ConvertigoTutorial** Public

Pin Unwatch 1 Fork 0 Star 0

master 1 branch 0 tags

Go to file Add file Code

**Commits**

Commit Message	Author	Time Ago	SHA-1
Nogaemi76 Added new test case	Nogaemi76	5 minutes ago	d55f188
DisplayObjects Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76	43 minutes ago	
_c8oProject Added new test case	Nogaemi76	5 minutes ago	
css Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76	43 minutes ago	
js Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76	43 minutes ago	
.gitattributes Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76	43 minutes ago	
.gitignore Added .gitignore	Nogaemi76	31 minutes ago	
c8oProject.yaml Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76	43 minutes ago	
index.html Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76	43 minutes ago	
project.md Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76	43 minutes ago	
readme.md Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76	43 minutes ago	

**About**

No description, website, or topics provided.

Readme Activity 0 stars 1 watching 0 forks

**Releases**

No releases published [Create a new release](#)

**Packages**

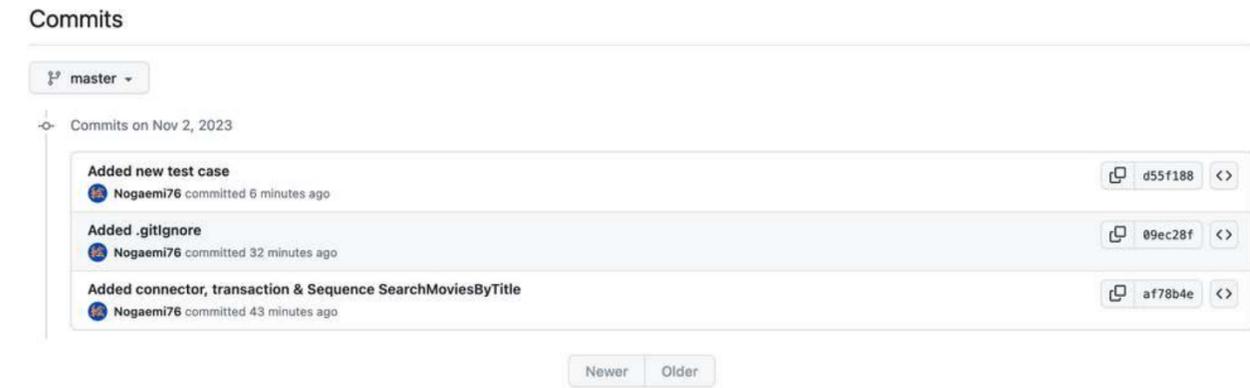
No packages published [Publish your first package](#)

**Languages**

- JavaScript 91.6%
- CSS 5.0%
- HTML 3.4%

**readme.md**

This convertigo project template can be used to start a Mobile Builder Ionic project with Convertigo.



**Commits**

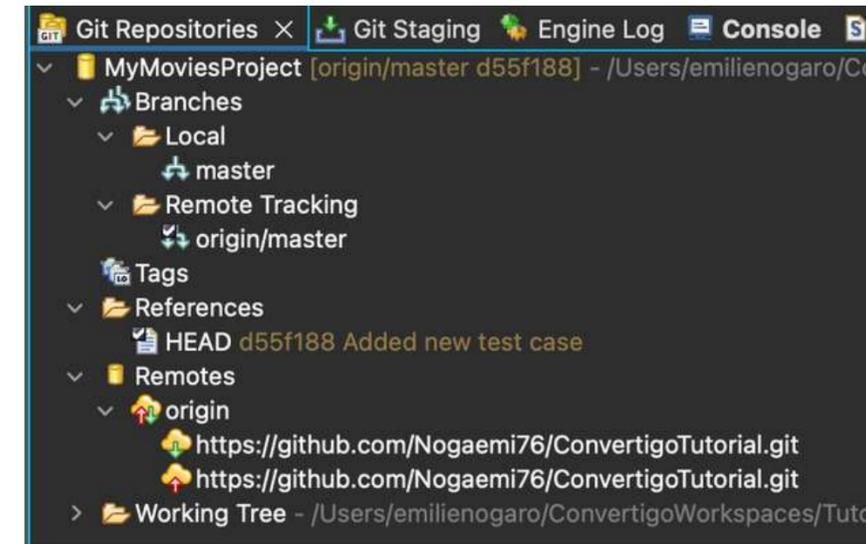
master

Commits on Nov 2, 2023

Added new test case	Nogaemi76 committed 6 minutes ago	d55f188
Added .gitignore	Nogaemi76 committed 32 minutes ago	09ec28f
Added connector, transaction & Sequence SearchMoviesByTitle	Nogaemi76 committed 43 minutes ago	af78b4e

Newer Older

The **Remote branch** appears in the **Git Repository view**.



Git Repositories Git Staging Engine Log Console

MyMoviesProject [origin/master d55f188] - /Users/emilienogaro/Co

- Branches
  - Local
    - master
  - Remote Tracking
    - origin/master
- Tags
- References
  - HEAD d55f188 Added new test case
- Remotes
  - origin
    - https://github.com/Nogaemi76/ConvertigoTutorial.git
    - https://github.com/Nogaemi76/ConvertigoTutorial.git
- Working Tree - /Users/emilienogaro/ConvertigoWorkspaces/Tuto



# 7.7 Clone a project

Let's say you want to clone a project in your studio.

For example, you want to use the **library lib\_UserManager developed by Convertigo**.

It is used to **include user management and authentication in a Convertigo project**.

## lib\_UserManager

### User management and Authentication for your projects

The lib\_UserManager enables your projects to include user management and authentication in your apps. This library will handle :

- user login with user/password using a salted password security
- user login using OpenID (Google, Azure & linkedin)

When using user/password, the library will use the **lib\_usermanager\_fullsync** database to store userids and salted/hashed password

You can find the repository in GitHub :

<https://github.com/convertigo/c8oprj-lib-user-manager>



# 7.7 Clone a project

As explained in the ReadMe of lib\_UserManager in GitHub, the **simplest way** to clone a project is

- **NOT** by using the Git Repositories view (more complex eclipse-based process).
- by using the **Convertigo project import Wizard** in the **Project view** (customized process developed by Convertigo).

Copy the project url from the ReadMe of the repo in GitHub :

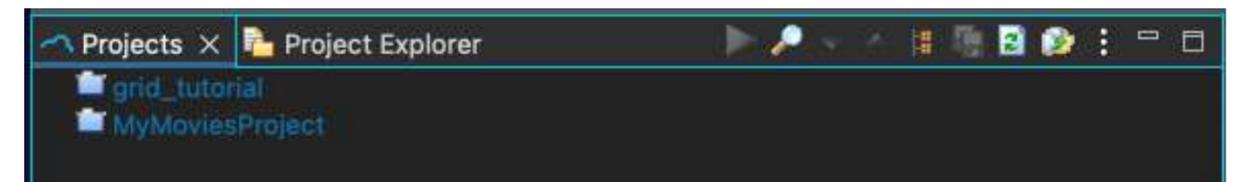
lib\_UserManager=https://github.com/convertigo/c8oprj-lib-user-manager/archive/8.0.X.zip

Usage	Click the copy button
To contribute	lib_UserManager=https://github.com/convertigo/c8oprj-lib-user-manager.git:branch=f
To simply use	lib_UserManager=https://github.com/convertigo/c8oprj-lib-user-manager/archive/8.0.X.

 lib\_UserManager=https://github.com/convertigo/c8oprj-lib-user-manager/archive/8.0.X. 

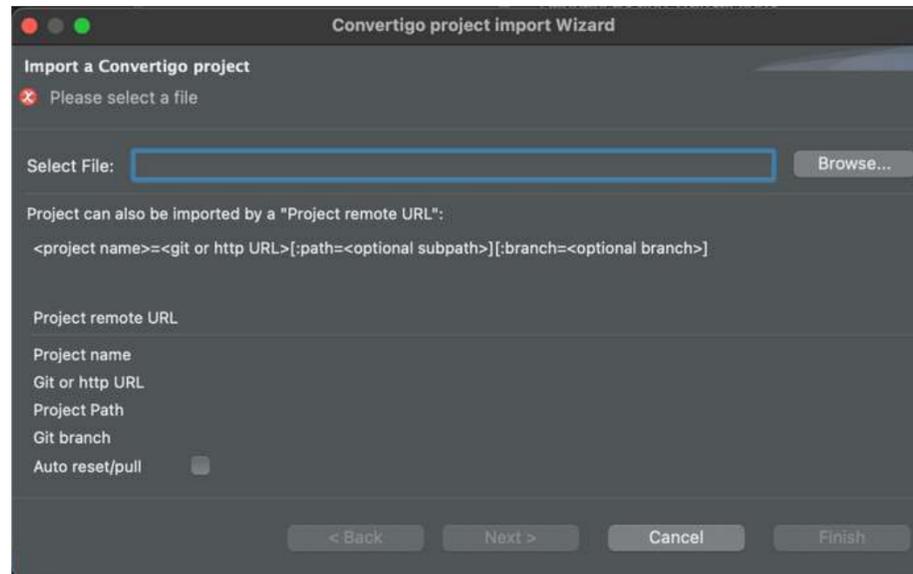


Click on the **Import a project in treeview** button to open the **Convertigo project import Wizard**.

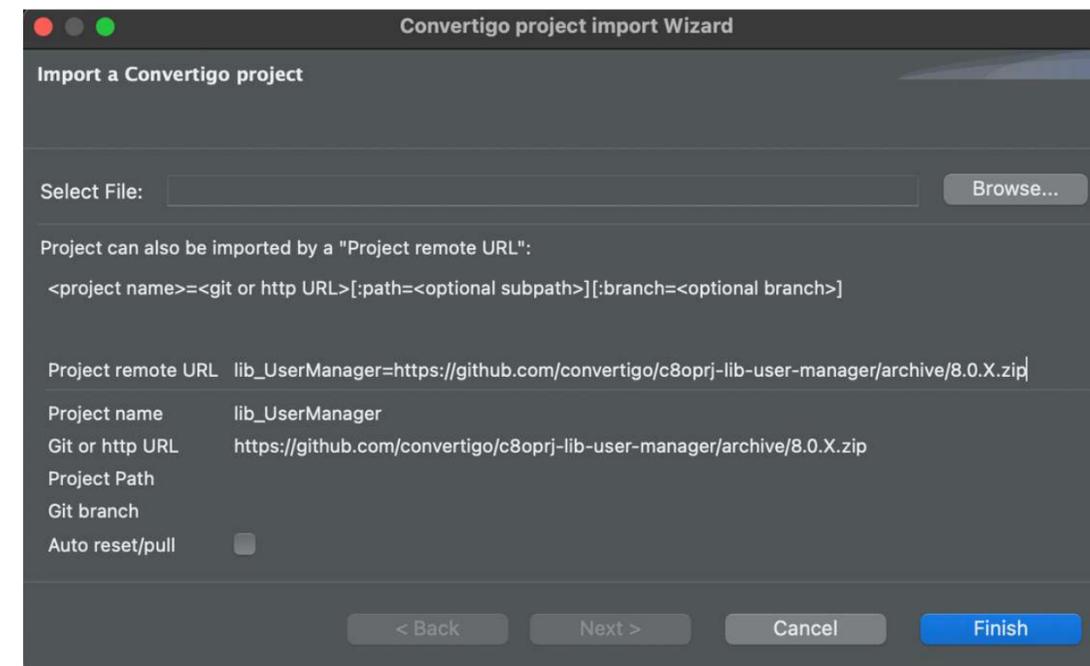


# 7.7 Clone a project

The **Convertigo** project import Wizard opens.



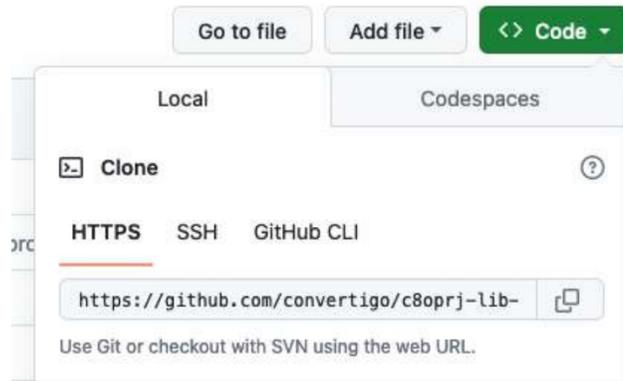
Paste the **project url** in the **Project remote URL** field and click on **Finish**.



```
Project remote URL lib_UserManager=https://github.com/convertigo/c8oprj-lib-user-manager/archive/8.0.X.zip
Project name lib_UserManager
Git or http URL https://github.com/convertigo/c8oprj-lib-user-manager/archive/8.0.X.zip
```



# 7.7 Clone a project

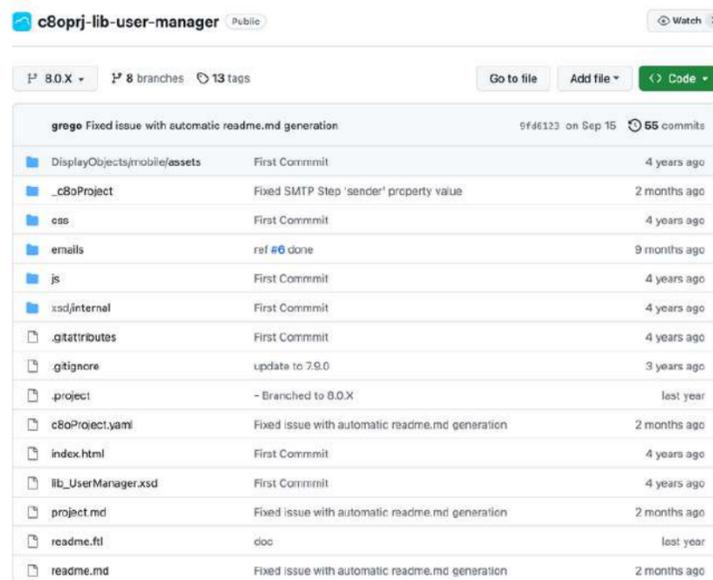


## Important :

Usually, when **cloning a GitHub repo**, you copy it from the usual repo url and the project name is not already present in the url.

In that case, you have to **include it manually in the project name field**.

To find the **project name of a Convertigo library**, go to the **c8oProject.yaml** file in GitHub.



The project name (lib\_UserManager) is indicated at the very beginning of the file.

[c8oprj-lib-user-manager / c8oProject.yaml](#)

```
Code Blame 54 lines (48 loc) · 3.69 KB
1  rconvertigo: 8.0.0.m006
2  ↓lib_UserManager [core.Project]:
3    comment: |
4      '# User management and Authentication for your projects
5
6      The lib_UserManager enables your projects to include user management and authentication in your apps. This library will handle :
7
8      - user login with user/password using a salted password security
9      - user login using OpenID (Google, Azure & linkedin)
10
```

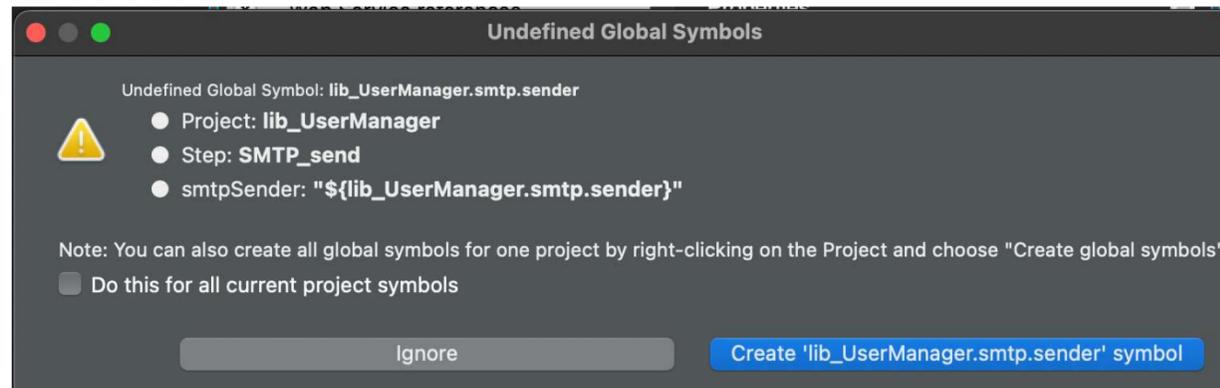


↓ lib\_UserManager



# 7.7 Clone a project

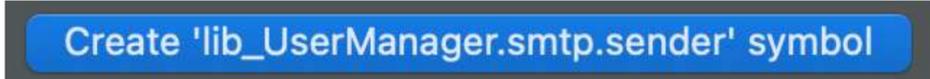
If the cloned project has symbols, the **Undefined Global Symbol** window appears.



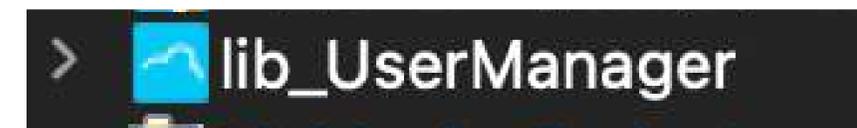
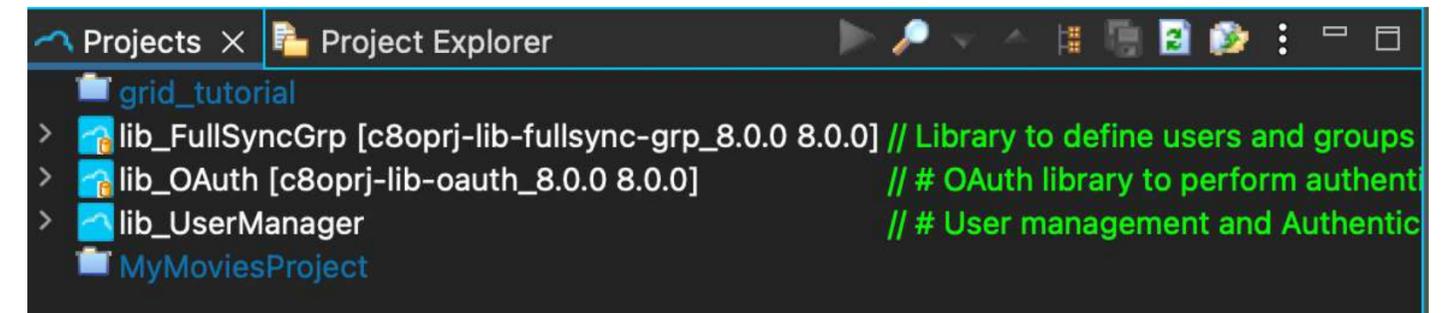
Select **Do this for all current symbols**.



Click on Create 'XXX' symbol ('XXX' depends on the symbol name).



The **library is imported** and appears in the **Projects view**.



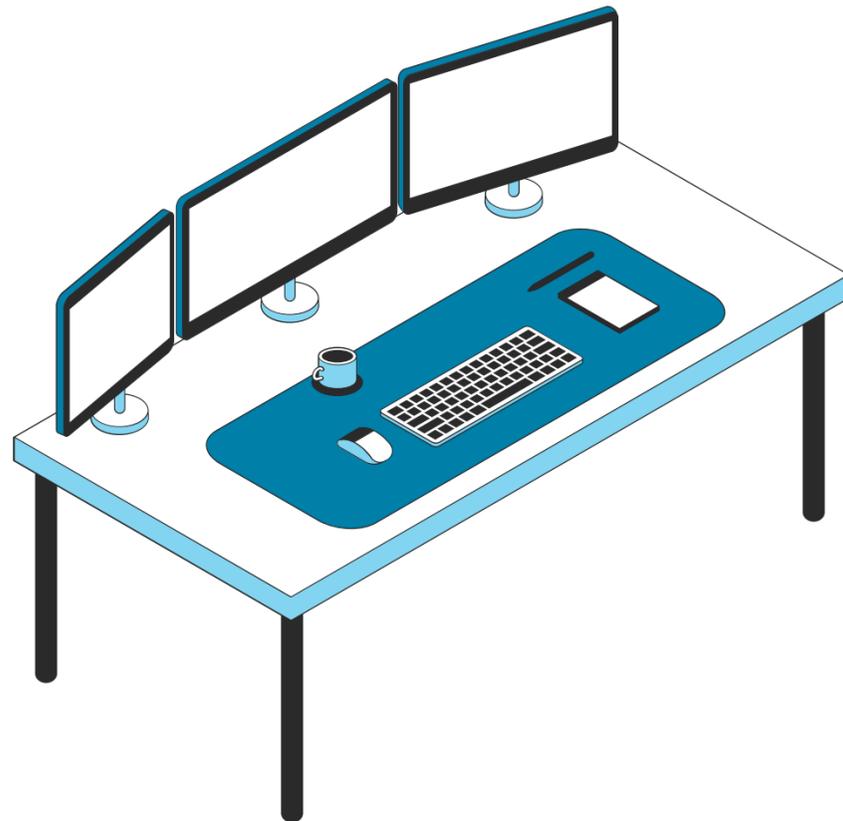
Important :

The library lib\_UserManager **uses other libraries** (as shown in References folder) and they were imported as well.



# 8 – Test platform

How to test your backend.



**8.1** Access the Test platform

---

**8.2** Test a transaction

---

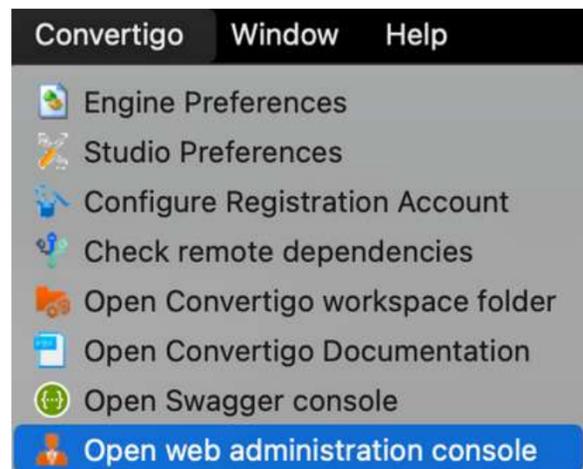
**8.3** Test a sequence

# 8.1 Access the Test platform

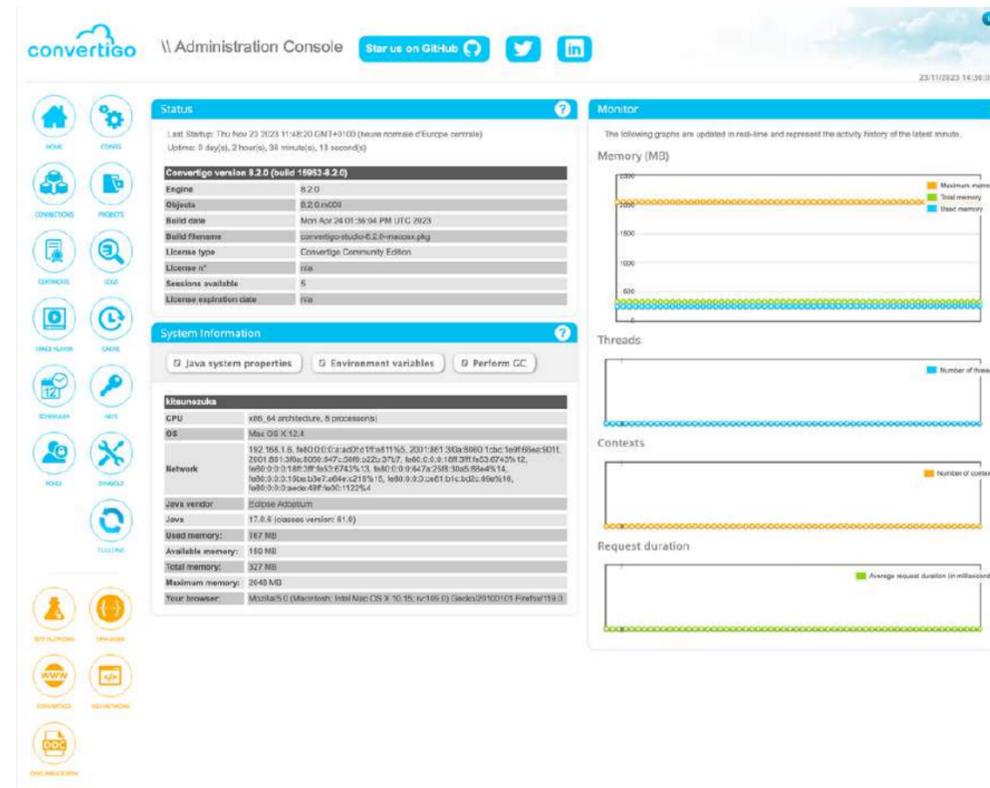
Convertigo provides a Test platform to test your backend and your frontend.

To access the Test platform

Open the **web administration console**.



In the web administration console



Click on the **Test platform icon**.



TEST PLATFORM

# 8.1 Access the Test platform

In the Test platform are displayed **all the projects of your workspace.**

WELCOME ON CONVERTIGO LOW CODE PLATFORM  
Test Platform

Convertigo version: 8.2.0 (build 15952-8.2.0)  
Engine version: 8.2.0  
Objects version: 8.2.0.m006  
Java version: 17.0.6  
Classes version: 61.0 Eclipse Adoptium

Project name	Comment	Deployment date	Test platform	Web-service definition
lib_FullSyncGrp (8.0.0)	Library to define users and groups for fullsync replication filtering	n/a	+	wdl
lib_OAuth (1.3.0)	# OAuth library to perform authentication This is the OAuth Library for Convertigo applications. Thi...	n/a	+	wdl
lib_UserManager (2.0.18)	# User management and Authentication for your projects The lib_UserManager enables your projects to...	n/a	+	wdl
MyMoviesProject (V1.1)	Convertigo NGX builder Project	6 nov. 2023 11:21	+	wdl



Click on MyMoviesProject to select it.



MyMoviesProject (V1.1)

Convertigo NGX builder Project

# 8.1 Access the Test platform

In the MyMoviesProject page of the Test platform, we can see **all the transactions and sequences** of the project.

The screenshot shows the Convertigo Test Platform interface. At the top, there's a navigation bar with the Convertigo logo, social media links, and a 'Welcome admin' message. Below this is a large blue banner that reads 'WELCOME ON CONVERTIGO LOW CODE PLATFORM' and 'Test Platform'. The main content area is titled 'MyMoviesProject project (V1.1)' and 'Convertigo NGX builder Project'. It features a sidebar with navigation icons for Admin Console, Projects, Swagger, Documentation, Convertigo, and DevNetwork. The main panel is divided into sections: 'Mobile application' with a table of application details (name, endpoint, ID, version), 'PWA / Web App' with a QR code and 'Execute here' button, and a list of connectors and sequences: 'HTTP\_connector\_MyMoviesProject connector', 'void connector', and 'Sequence (s)'. At the bottom, there are 'Deploy all' and 'Collapse all' buttons.

This block shows a zoomed-in view of the project and connector list. It features a magnifying glass icon next to the text 'MyMoviesProject project' and 'Convertigo NGX builder Project'. Below this, there are three blue buttons with magnifying glass icons and right-pointing arrows: 'HTTP\_connector\_MyMoviesProject connector', 'void connector', and 'Sequence (s)'. At the bottom right, there are 'Deploy all' and 'Collapse all' buttons.



This block shows a zoomed-in view of the connector and sequence list. It features a blue button with a downward-pointing arrow and the text 'HTTP\_connector\_MyMoviesProject connector'. Below this, there are two blue buttons with right-pointing arrows: 'Default\_transaction' and 'SearchMoviesByTitle'. Below these, there are two more blue buttons with right-pointing arrows: 'void connector' and 'Sequence (s)'. At the bottom, there is a blue button with a right-pointing arrow and the text 'SearchMoviesByTitle'.

# 8.2 Test a transaction

Let's test our SearchMoviesByTitle transaction.

When we deploy the transaction tab, we can see 2 parts:

The screenshot shows the 'SearchMoviesByTitle' transaction editor. At the top, there is a blue header with the text 'HTTP\_connector\_MyMoviesProject connector'. Below this, there are two tabs: 'Default\_transaction' and 'SearchMoviesByTitle'. The 'SearchMoviesByTitle' tab is active and shows two input fields: '\_\_header\_Authorization' with a dotted pattern and 'movieTitle' which is empty. To the right of each field is a checkbox labeled 'Send value', both of which are checked. Below the input fields is an 'Execute' button. Underneath the input fields, there is a section titled 'Test cases'. It contains a table with two columns: the first column lists the variables '\_\_header\_Authorization' and 'movieTitle', and the second column contains the values '\*\*\*\*\*' and 'avatar' respectively. Below the table are 'Edit' and 'Execute' buttons.

- an **editor with our transaction variables** where we can enter a movieTitle variable.

This close-up shows the input fields for the transaction editor. It features two rows: the first row has the label '\_\_header\_Authorization' followed by a text box containing a dotted pattern and a checked 'Send value' checkbox; the second row has the label 'movieTitle' followed by an empty text box and a checked 'Send value' checkbox. An 'Execute' button is located to the right of these fields.

- the **test case we created in our transaction.**

This close-up shows the 'Test cases' section of the editor. It features a table with the title 'Test cases' and a sub-title 'Test\_Case\_title\_avatar'. The table has two columns: the first column lists the variables '\_\_header\_Authorization' and 'movieTitle', and the second column contains the values '\*\*\*\*\*' and 'avatar' respectively. Below the table are 'Edit' and 'Execute' buttons.

## 8.2 Test a transaction

Let's try the **test case** we created in our project with "avatar" as value for the movieTitle variable.

**Test cases**

Test\_Case\_title\_avatar

__header_Authorization	*****
movieTitle	avatar



The result will be **displayed in XML** by default.

**Execution mode**

C80 lib  XML  Json  Binary



Click on **Execute** to run the test case.



The result is **displayed in XML**.

**Execution mode**

C80 lib  XML  Json  Binary

**Execution result**

Generated URL :  
[http://localhost:18080/convertigo/projects/MyMoviesProject/pxml?\\_\\_connector=HTTP\\_connector\\_MyMoviesProject&\\_\\_transaction=SearchMoviesByTitle&\\_\\_testcase=Test\\_Case\\_title\\_avatar&\\_\\_xsrfToken=fkb0YA-kZeRVMYoQza2ozQKuVRLPx6Qj-BEx-Snllko](http://localhost:18080/convertigo/projects/MyMoviesProject/pxml?__connector=HTTP_connector_MyMoviesProject&__transaction=SearchMoviesByTitle&__testcase=Test_Case_title_avatar&__xsrfToken=fkb0YA-kZeRVMYoQza2ozQKuVRLPx6Qj-BEx-Snllko)

```
<?xml version="1.0" encoding="UTF-8"?><document connector="HTTP_connector_MyMoviesProject"
context="studio_MyMoviesProject:C:HTTP_connector_MyMoviesProject"
contextId="studio_MyMoviesProject:C:HTTP_connector_MyMoviesProject" fromStub="false"
fromcache="false" generated="Thu Nov 23 18:16:36 CET 2023" project="MyMoviesProject"
sequence="" signature="1700759796279" transaction="SearchMoviesByTitle" version="8.2.0 (build
15952-8.2.0)">
  <object type="object">
    <page type="integer">1</page>
    <results length="20" type="array">
      <object type="object">
        <adult type="boolean">>false</adult>
        <backdrop_path type="string">/vL5LR6WdxWPjLpFRLe133jXWsh5.jpg</backdrop_path>
        <genre_ids length="4" type="array">
          <value type="integer">28</value>
          <value type="integer">12</value>
          <value type="integer">14</value>
          <value type="integer">878</value>
        </genre_ids>
        <id type="integer">19995</id>
        <original_language type="string">en</original_language>
        <original_title type="string">Avatar</original_title>
        <overview type="string">Un marine paraplégique, envoyé sur la lune Pandora
pour une mission unique, est tiraillé entre suivre ses ordres et protéger le monde qu'il
considère dorénavant comme le sien.</overview>
        <popularity type="double">142.46</popularity>
        <poster_path type="string">/3npygfmEhqnmNTmDWhHLz1LPcbA.jpg</poster_path>
        <release_date type="string">2009-12-15</release_date>
        <title type="string">Avatar</title>
        <video type="boolean">>false</video>
        <vote_average type="double">7.575</vote_average>
        <vote_count type="integer">30029</vote_count>
      </object>
      <object type="object">
        <adult type="boolean">>false</adult>
        <backdrop_path type="string">/8rpDcsfLJypb06vREc0547VKqEv.jpg</backdrop_path>
        <genre_ids length="3" type="array">
          <value type="integer">878</value>
          <value type="integer">12</value>
```



## 8.2 Test a transaction

Let's try the editor with our transaction variables with "titanic" as value for the movieTitle variable.



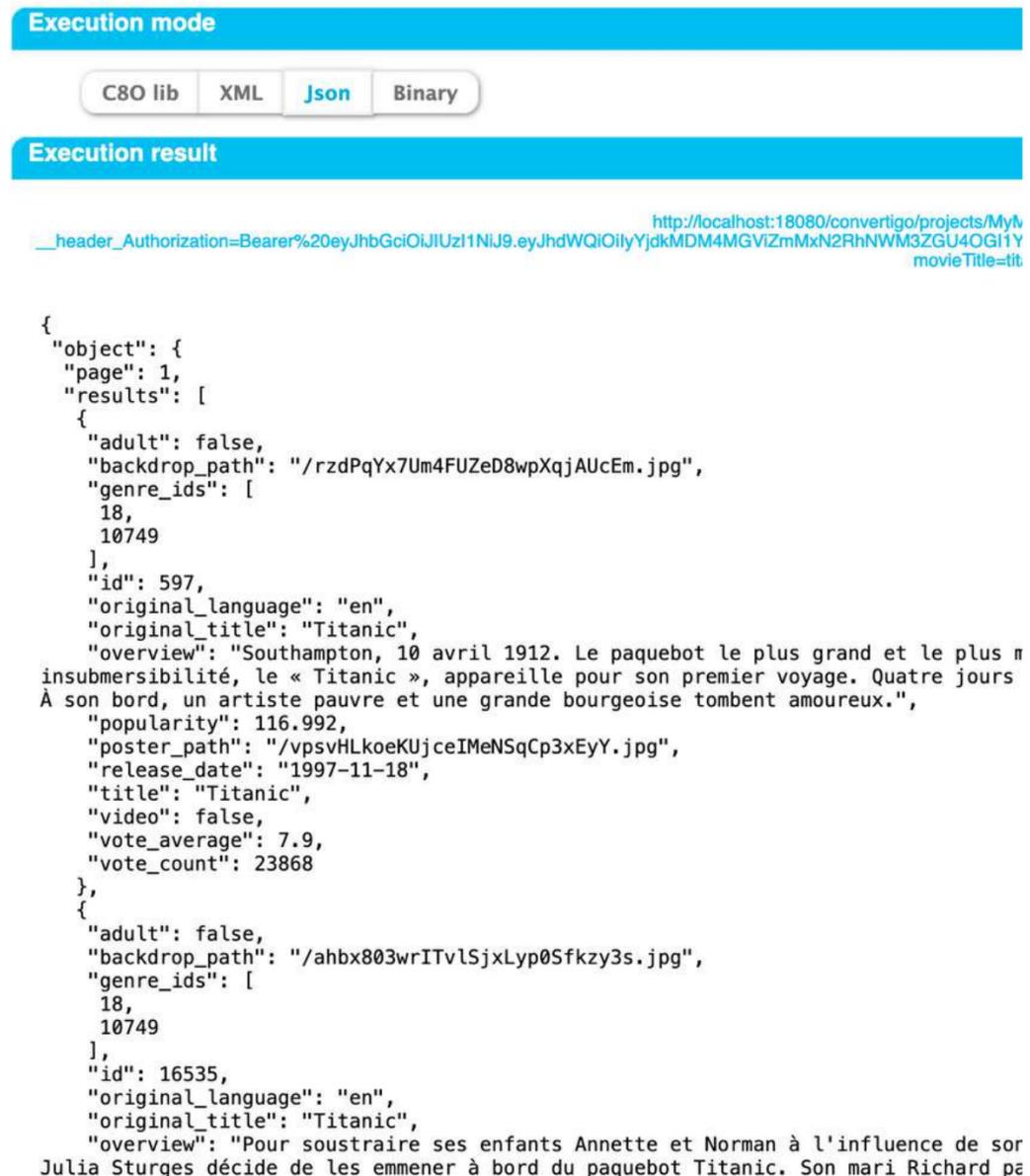
Let's change the Execution mode to Json.



Click on Execute to run the test case.



The result is displayed in JSON.



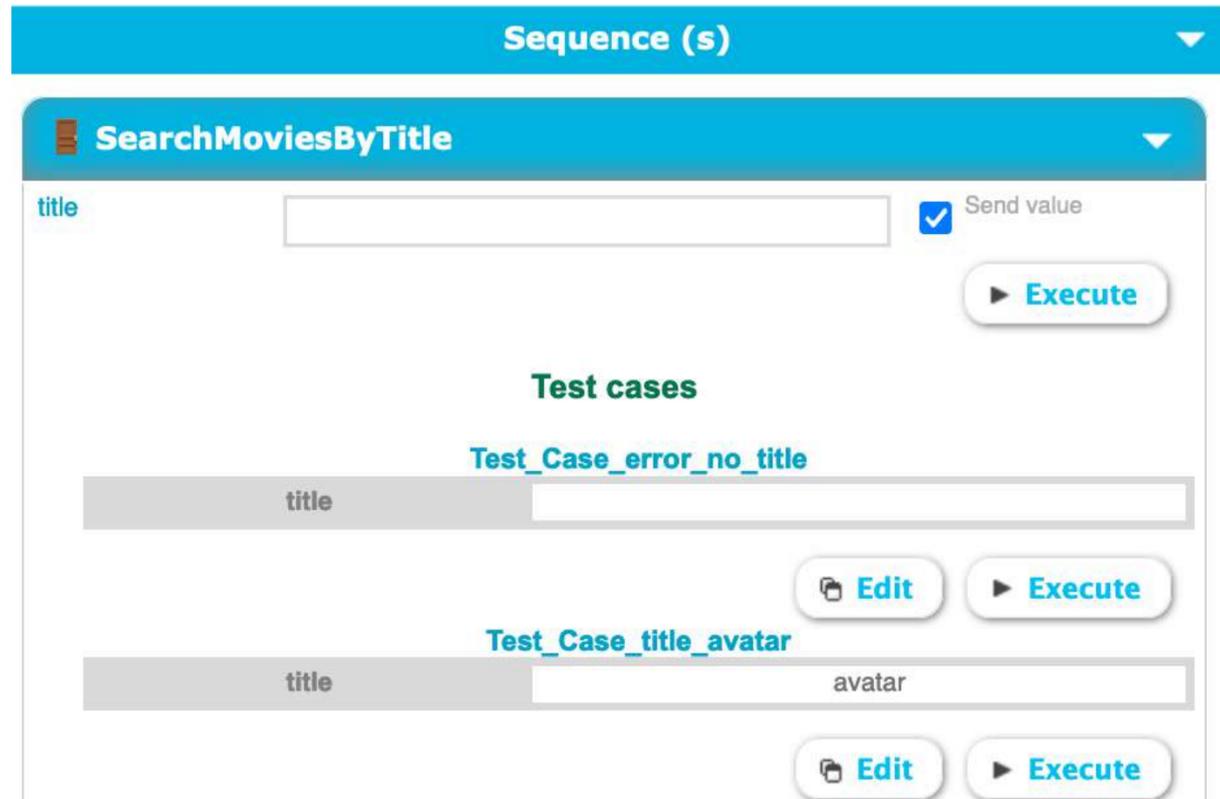
```
{
  "object": {
    "page": 1,
    "results": [
      {
        "adult": false,
        "backdrop_path": "/rzdPqYx7Um4FUZeD8wpXqjAUcEm.jpg",
        "genre_ids": [
          18,
          10749
        ],
        "id": 597,
        "original_language": "en",
        "original_title": "Titanic",
        "overview": "Southampton, 10 avril 1912. Le paquebot le plus grand et le plus insubmersibilité, le « Titanic », appareille pour son premier voyage. Quatre jours à son bord, un artiste pauvre et une grande bourgeoise tombent amoureux.",
        "popularity": 116.992,
        "poster_path": "/vpsvHLkoeKUjceIMeNSqCp3xEyY.jpg",
        "release_date": "1997-11-18",
        "title": "Titanic",
        "video": false,
        "vote_average": 7.9,
        "vote_count": 23868
      },
      {
        "adult": false,
        "backdrop_path": "/ahbx803wrITvLSjxLyp0Sfky3s.jpg",
        "genre_ids": [
          18,
          10749
        ],
        "id": 16535,
        "original_language": "en",
        "original_title": "Titanic",
        "overview": "Pour soustraire ses enfants Annette et Norman à l'influence de sa mère, Julia Sturges décide de les emmener à bord du paquebot Titanic. Son mari Richard pa
```



# 8.3 Test a sequence

Now, let's test our SearchMoviesByTitle sequence.

Here again, we can see 2 parts:



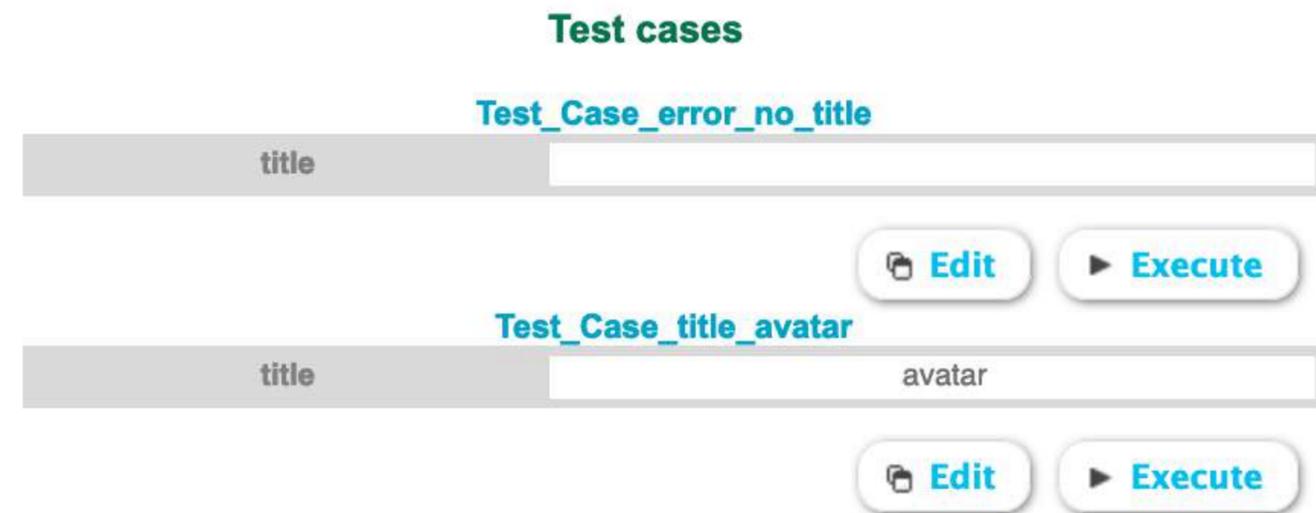
The screenshot shows a sequence editor for 'SearchMoviesByTitle'. At the top, there is a dropdown menu labeled 'Sequence (s)' and another labeled 'SearchMoviesByTitle'. Below the 'SearchMoviesByTitle' dropdown, there is a text input field labeled 'title', a checked checkbox for 'Send value', and an 'Execute' button. Below this, there is a section titled 'Test cases'. It contains two test case entries: 'Test\_Case\_error\_no\_title' and 'Test\_Case\_title\_avatar'. Each entry has a 'title' input field and an 'Execute' button. The 'Test\_Case\_title\_avatar' entry also has an 'avatar' input field.

- an editor with our sequence variable where we can enter a title variable.



This is a close-up of the 'SearchMoviesByTitle' editor. It shows the dropdown menu with the text 'SearchMoviesByTitle'. Below it is a text input field labeled 'title', a checked checkbox for 'Send value', and an 'Execute' button.

- the test cases we created in our transaction.



This is a close-up of the 'Test cases' section. It shows two test case entries: 'Test\_Case\_error\_no\_title' and 'Test\_Case\_title\_avatar'. Each entry has a 'title' input field and an 'Execute' button. The 'Test\_Case\_title\_avatar' entry also has an 'avatar' input field.

## 8.3 Test a sequence

Let's try the error test case we created in our project with no value for the title variable.

**Test cases**

**Test\_Case\_error\_no\_title**

title

 **Edit**  **Execute**



The **Execution mode** is in **XML**.

**Execution mode**

**C80 lib** **XML** **Json** **Binary**



Click on **Execute** to run the test case.

 **Execute**

The result is **displayed in XML**.

```
Execution result
Generated URL :
http://localhost:18080/convertigo/projects/MyMoviesProject/pxml?__sequence=SearchMoviesByTitle&__testcase=Test_Case_error_no_title&__xsrfToken=fkb0YA-kZeRVMYoQza2ozQKuVRLPx6Qj-BEx-Snllko-

<?xml version="1.0" encoding="UTF-8"?><document connector=""
context="studio_MyMoviesProject:S:SearchMoviesByTitle"
contextId="studio_MyMoviesProject:S:SearchMoviesByTitle" fromStub="false" fromcache="false"
generated="Thu Nov 23 18:18:34 CET 2023" project="MyMoviesProject"
sequence="SearchMoviesByTitle" signature="1700759914675" transaction="" version="8.2.0 (build
15952-8.2.0)">
  <error project="MyMoviesProject" sequence="SearchMoviesByTitle" type="project">
    <code>-1</code>
    <message>An unexpected error has occurred while the execution of the requested object
'SearchMoviesByTitle'.</message>
    <details>Cannot invoke "String.indexOf(int)" because "s" is null</details>
    <context/>
    <exception/>
    <stacktrace/>
  </error>
</document><!--
Generated by Convertigo Enterprise Mobility Server
Requester: XmlServletRequester
-->
```

Let's change the **Execution mode** to **Json**, and execute the test again.

The result is **displayed in JSON**.

```
Execution mode
C80 lib XML Json Binary Fullscreen Reload execution result frame

Execution result
Generated URL :
http://localhost:18080/convertigo/projects/MyMoviesProject/json?__sequence=SearchMoviesByTitle&__testcase=Test_Case_error_no_title&__xsrfToken=fkb0YA-kZeRVMYoQza2ozQKuVRLPx6Qj-BEx-Snllko-

{
  "error": {
    "code": "-1",
    "message": "An unexpected error has occurred while the execution of the requested object
'SearchMoviesByTitle'.",
    "details": "Cannot invoke \"String.indexOf(int)\" because \"s\" is null",
    "context": "",
    "exception": "",
    "stacktrace": "",
    "attr": {
      "project": "MyMoviesProject",
      "sequence": "SearchMoviesByTitle",
      "type": "project"
    }
  }
}
```

## 8.3 Test a sequence

Let's try the editor with our sequence variable with "titanic" as value for the title variable.



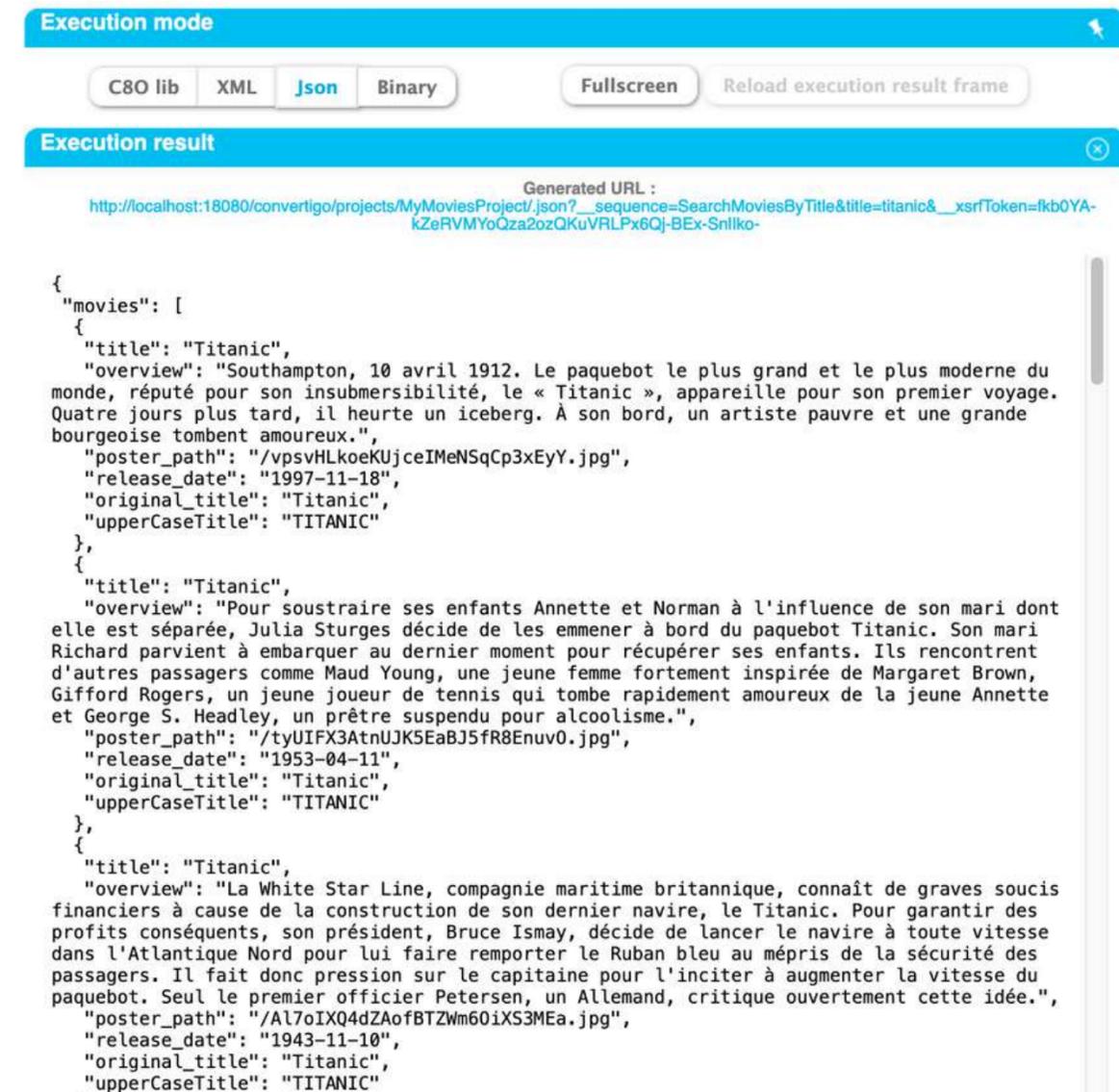
The Execution mode is in Json.



Click on Execute to run the test case.



The result is displayed in JSON.



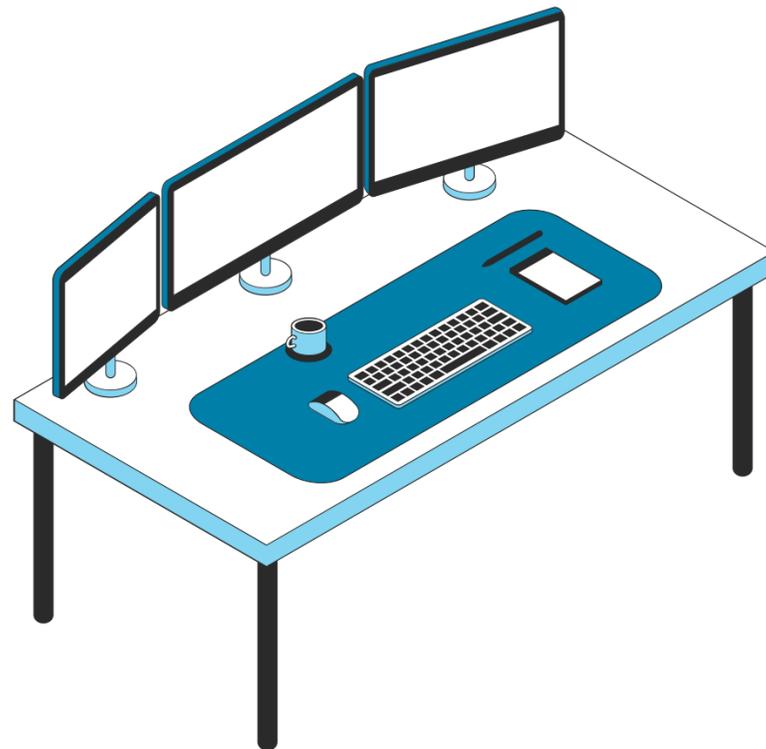
```
Generated URL :
http://localhost:18080/convertigo/projects/MyMoviesProject/json?__sequence=SearchMoviesByTitle&title=titanic&__xsrfToken=fbk0YA-
kZeRVMYoQza2ozQKuVRLPx6QJ-BEx-Snlko-

{
  "movies": [
    {
      "title": "Titanic",
      "overview": "Southampton, 10 avril 1912. Le paquebot le plus grand et le plus moderne du monde, réputé pour son insubmersibilité, le « Titanic », appareillé pour son premier voyage. Quatre jours plus tard, il heurte un iceberg. À son bord, un artiste pauvre et une grande bourgeoise tombent amoureux.",
      "poster_path": "/vpsvHLkoeKUjceIMeNSqCp3xEyY.jpg",
      "release_date": "1997-11-18",
      "original_title": "Titanic",
      "upperCaseTitle": "TITANIC"
    },
    {
      "title": "Titanic",
      "overview": "Pour soustraire ses enfants Annette et Norman à l'influence de son mari dont elle est séparée, Julia Sturges décide de les emmener à bord du paquebot Titanic. Son mari Richard parvient à embarquer au dernier moment pour récupérer ses enfants. Ils rencontrent d'autres passagers comme Maud Young, une jeune femme fortement inspirée de Margaret Brown, Gifford Rogers, un jeune joueur de tennis qui tombe rapidement amoureux de la jeune Annette et George S. Headley, un prêtre suspendu pour alcoolisme.",
      "poster_path": "/tyUIFX3AtnUJK5EaBJ5fR8Euv0.jpg",
      "release_date": "1953-04-11",
      "original_title": "Titanic",
      "upperCaseTitle": "TITANIC"
    },
    {
      "title": "Titanic",
      "overview": "La White Star Line, compagnie maritime britannique, connaît de graves soucis financiers à cause de la construction de son dernier navire, le Titanic. Pour garantir des profits conséquents, son président, Bruce Ismay, décide de lancer le navire à toute vitesse dans l'Atlantique Nord pour lui faire remporter le Ruban bleu au mépris de la sécurité des passagers. Il fait donc pression sur le capitaine pour l'inciter à augmenter la vitesse du paquebot. Seul le premier officier Petersen, un Allemand, critique ouvertement cette idée.",
      "poster_path": "/AL7oIXQ4dZAofBTZWm60iXS3MEa.jpg",
      "release_date": "1943-11-10",
      "original_title": "Titanic",
      "upperCaseTitle": "TITANIC"
    }
  ]
}
```



# 9 – URL mapper

How to expose an API REST.



**9.1** What is the URL mapper ?

---

**9.2** URL mapper steps

---

**9.3** Create an URL mapper for a transaction

---

**9.4** Test the URL mapper on Swagger

# 9.1 What is the URL mapper ?

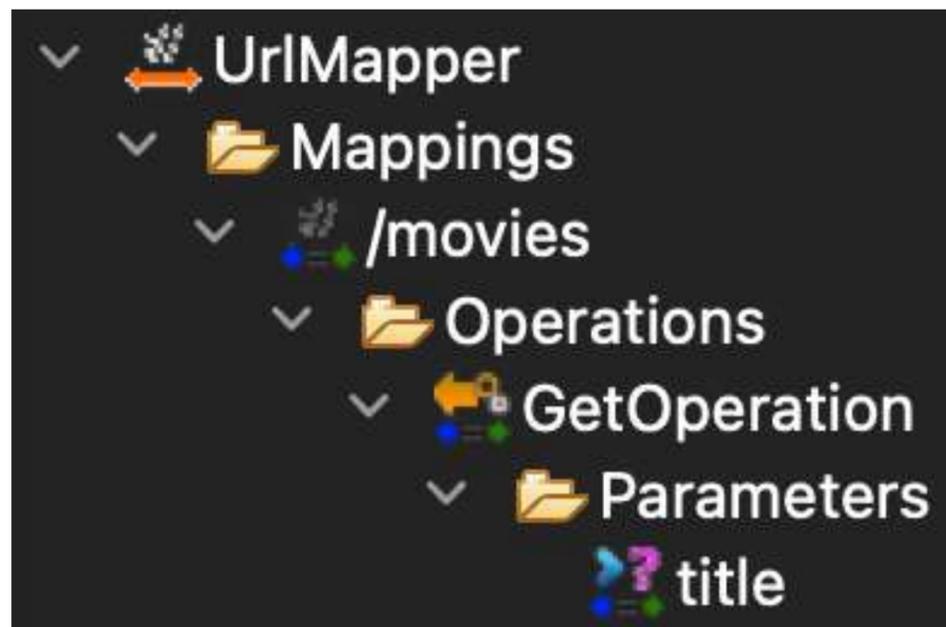
The **URL mapper** is able to map **RESTful urls** to **Convertigo requestables** such as **Sequences and Transactions**.

This way Convertigo **can expose RESTful APIs** to the outside world.

You can have **only one URLMapper per project**,

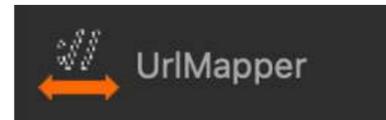
but an URLmapper **can map URLs to any other project deployed on the server**.

Example of URL mapper structure in a Convertigo project



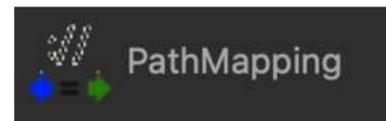
## 9.2 URL mapper steps

Convertigo provides steps to create the URL mapper.



### UrlMapper

This step defines the **URL mapper** to use in the project.



### PathMapping – Mapping step

This step defines a **mapping path associated with the mapper**, the **base URL structure** an API user will have to use **to access this API Service**.

For example: `/accounts/{accountid}`.

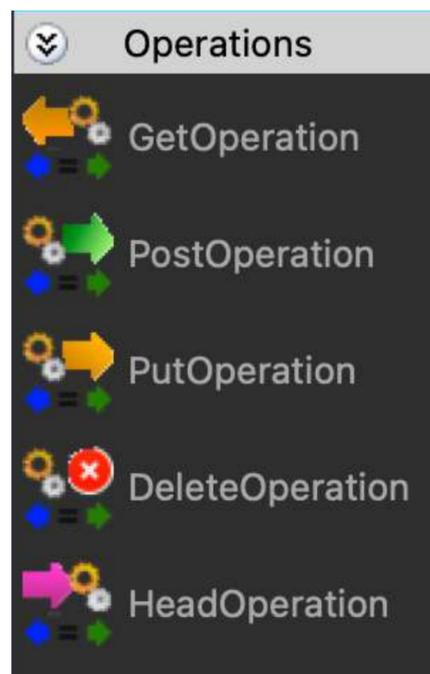


## 9.2 URL mapper Objects

### Operations Steps

These steps define the **HTTP operations associated with the mapping.**

For a **given operation on a given mapping,**  
you define here **what should be the Requestable (Sequence or Transaction) to be executed,**  
and **how will the variables for this requestable will be mapped.**



=> HTTP **GET** operation

=> HTTP **POST** operation

=> HTTP **PUT** operation

=> HTTP **DELETE** operation

=> HTTP **HEAD** operation



## 9.2 URL mapper Objects

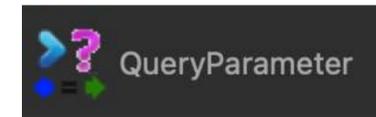
### Parameters Steps

Convertigo provides steps to define parameters associated with the operation.



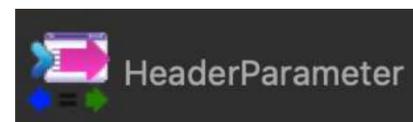
#### **PathParameter – Parameters step**

This step defines a path parameter by extracting the variable value from a segment of the URL path between {}. ex: /accounts/{accountid}



#### **QueryParameter – Parameters step**

This step defines a query parameter by extracting the variable value from the query string.  
ex: /accounts?verbose=1



#### **HeaderParameter – Parameters step**

This step defines a header parameter by extracting the variable value from the HTTP Header of this parameter name.



## 9.2 URL mapper Objects

### Responses Step

OperationResponse

#### OperationResponse – Responses step

This step defines an **HTTP response associated with the operation.**

When a service is invoked, it **responds with a HTTP status code.**

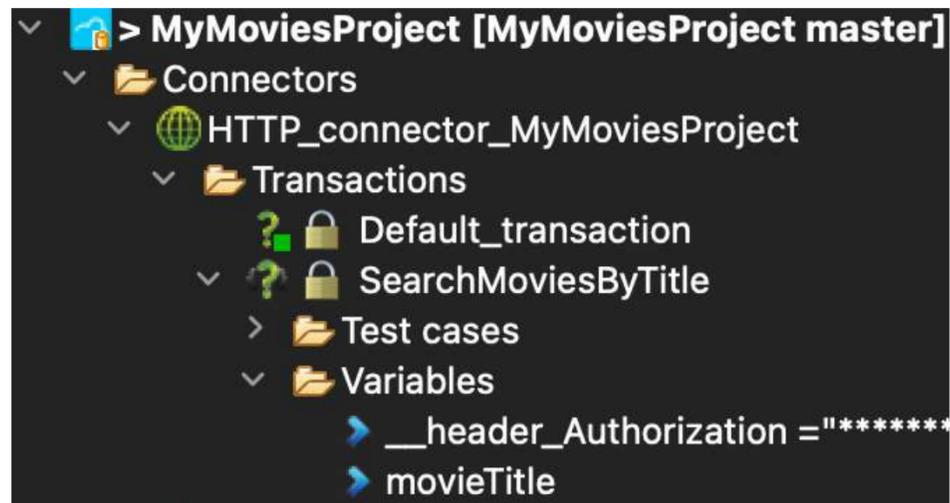
This mapping object will help you **define status codes** such as 200, 401 or any other **according to XPath resolution** done on a **Convertigo Sequence response.**

The Sequence response will be **scanned by all the UrlMappingResponse objects** defined for a given operation. The **first one having its XPath matching** will **generate the corresponding status code.**

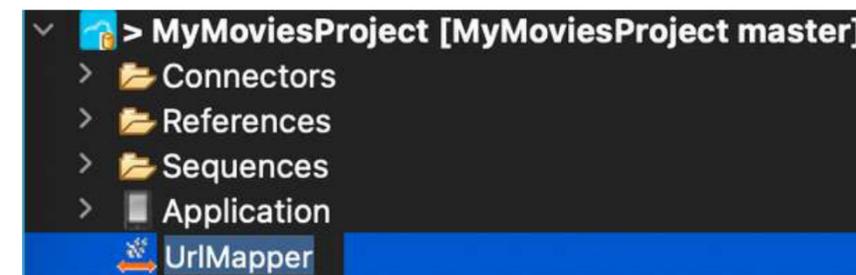
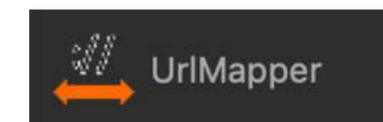
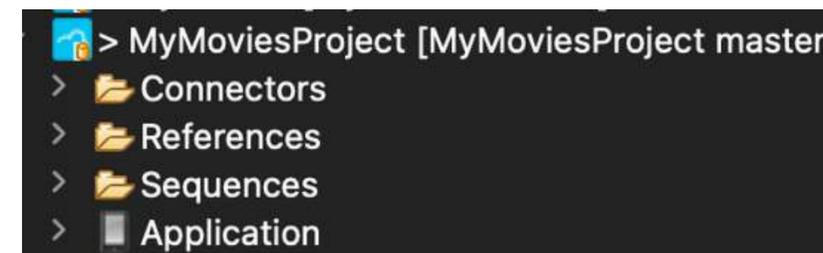


## 9.3 Create an URL mapper for a transaction

In our project, we have a **SearchMoviesByTitle** transaction, with a **variable** named **movieTitle**.  
Let's create an URL mapper for this transaction.

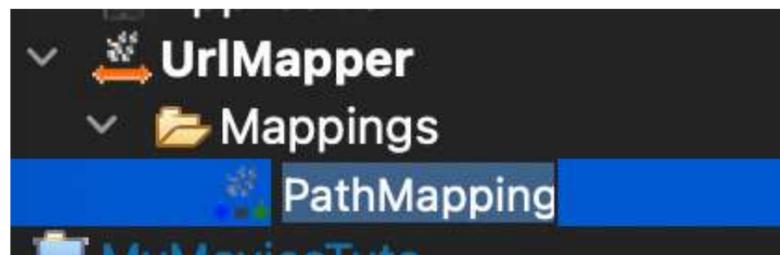
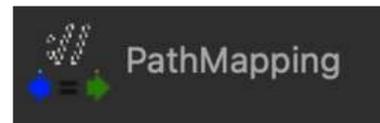
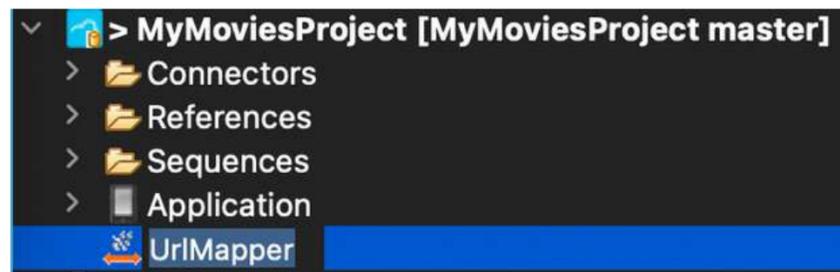


Drag and drop the **UrlMapper** step from the palette in the project.

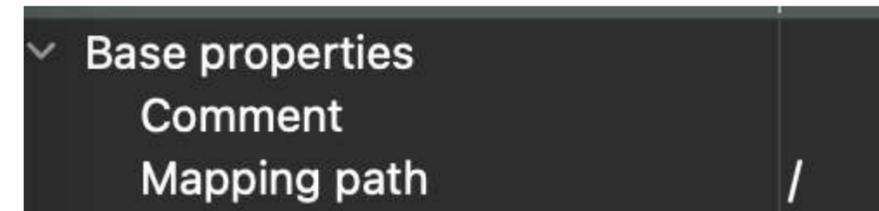


## 9.3 Create an URL mapper for a transaction.

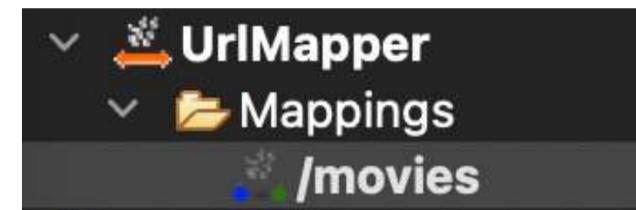
Drag and drop a **PathMapping** step from the palette in the **UrlMapper** step.



In the properties, rename the Mapping path as **/movies**.



Mapping path /movies

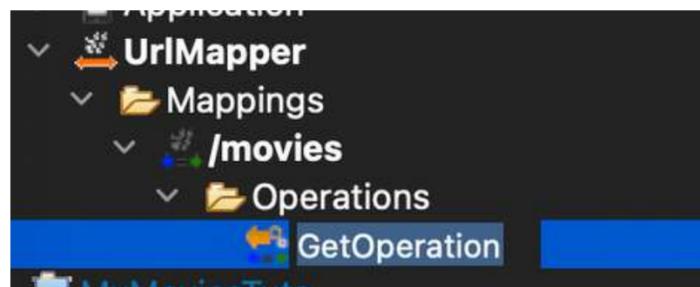
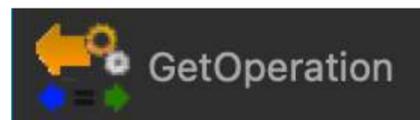
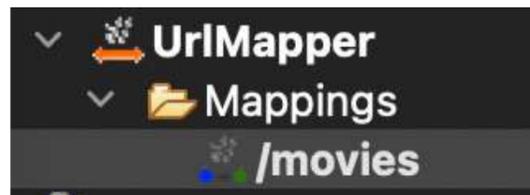


The path will appear as **/movies** in the url.

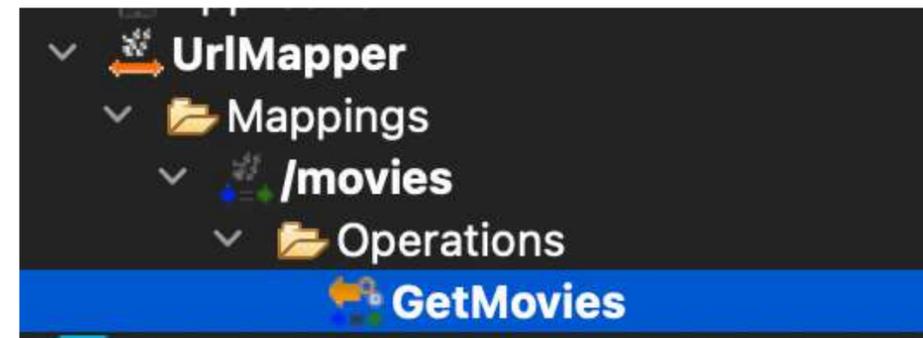


## 9.3 Create an URL mapper for a transaction.

Drag and drop a **GetOperation** step from the palette in the **PathMapping /movies** step.



Rename the GetOperation step as **GetMovies**.



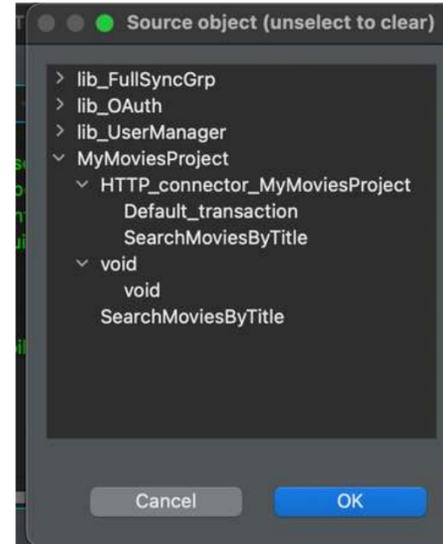
# 9.3 Create an URL mapper for a transaction.

Now, let's select which transaction or sequence we are going to map.

In the properties of GetMovies

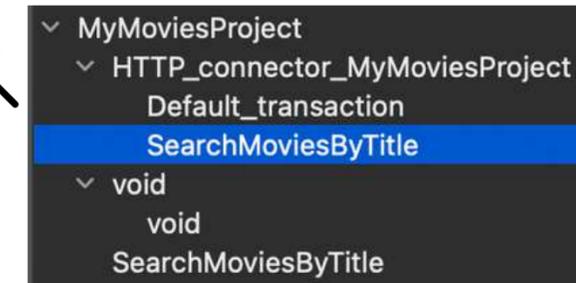


At the end of the line of the **Target requestable** property, click on this icon.

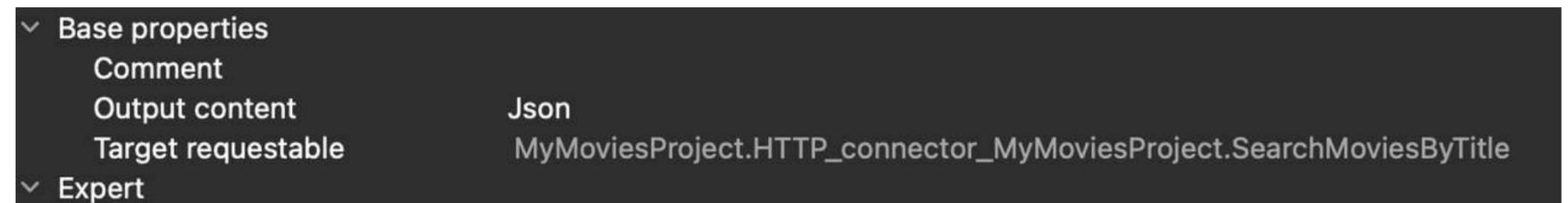


The **Source object** window appears.

Select the **SearchMoviesByTitle** transaction.

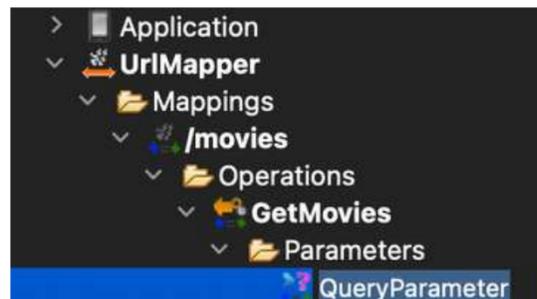
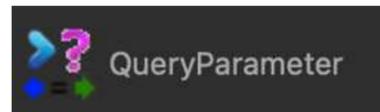
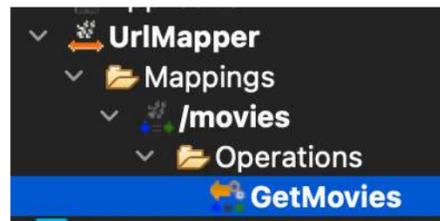


The SearchMoviesByTitle transaction appears as value in the Target requestable property of GetMovies.

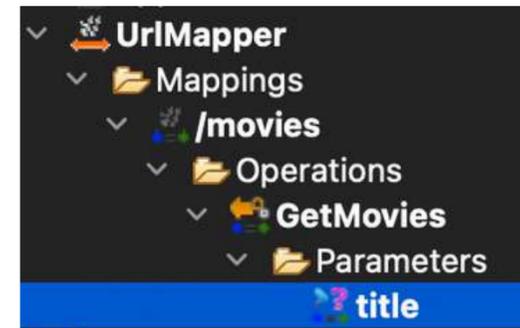


# 9.3 Create an URL mapper for a transaction.

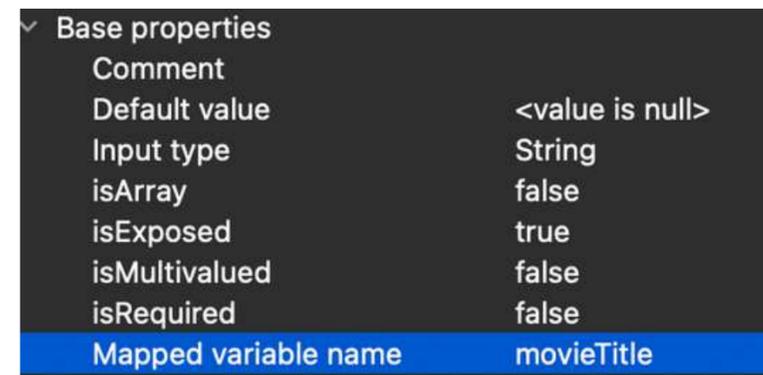
Drag and drop a **QueryParameter** step from the palette in the **GetMovies** step.



Rename the QueryParameter step as **title**.



In the **properties** of the QueryParameter, enter **movieTitle** (transaction variable name) as value of **Mapped variable name**.



# 9.4 Test the URL mapper on Swagger

Now, let's test our URL mapper on Swagger.

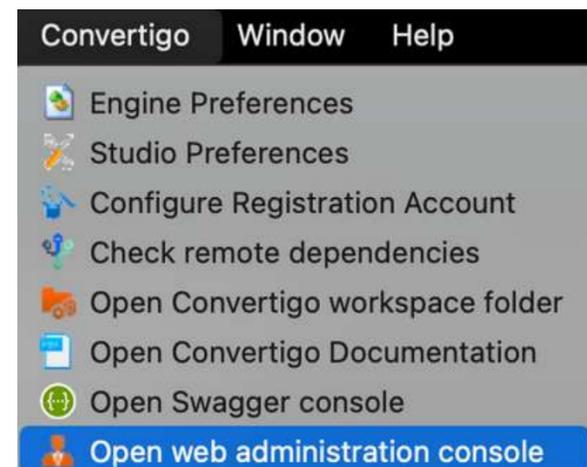
To open the Swagger console in your browser.



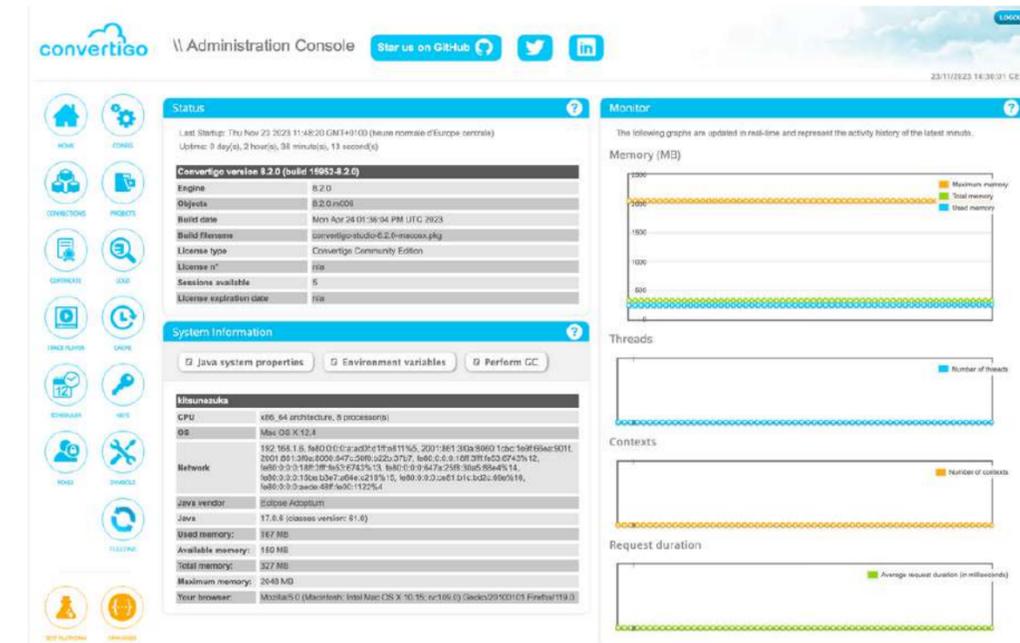
Click on  
Open Swagger console.



Or open the web  
administration console.



In the web administration console

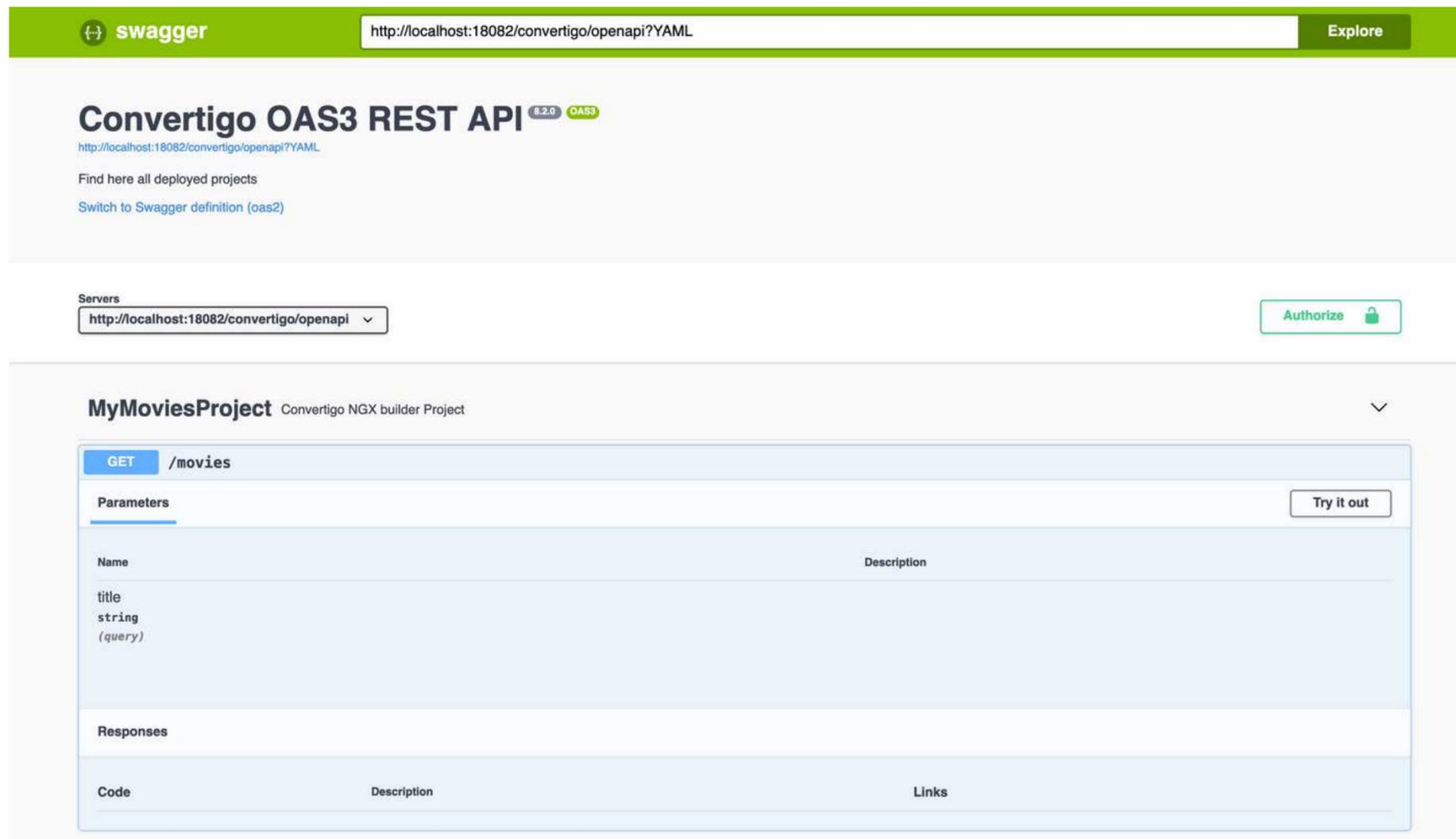


Click on the **Swagger icon**.



# 9.4 Test the URL mapper on Swagger

In the Swagger console of your browser, we can see a GET /movies request with a title parameter.



The screenshot shows the Swagger console interface. At the top, there is a green header with the Swagger logo and the URL `http://localhost:18082/convertigo/openapi?YAML`. Below this, the API is identified as "Convertigo OAS3 REST API" with version "3.2.0" and "OAS3" specification. The console shows a "Servers" dropdown menu with the selected server `http://localhost:18082/convertigo/openapi` and an "Authorize" button. The main content area displays the "MyMoviesProject" endpoint `GET /movies`. Under the "Parameters" section, a table lists the parameters:

Name	Description
title string (query)	

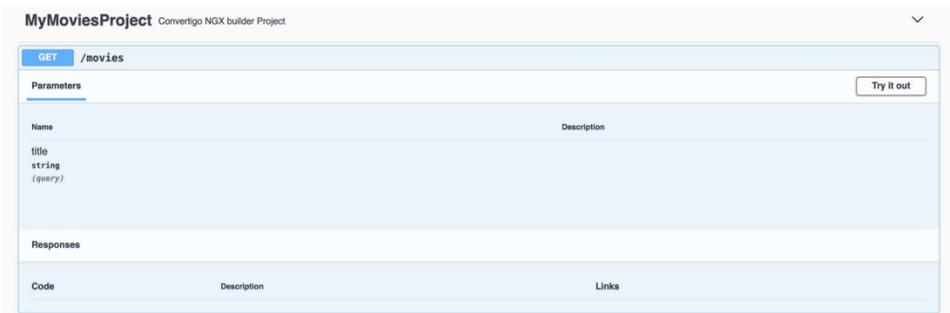
Below the parameters section, there is a "Responses" section with a table structure:

Code	Description	Links
------	-------------	-------



# 9.4 Test the URL mapper on Swagger

Let's test the GET /movies request with **Try it out**.

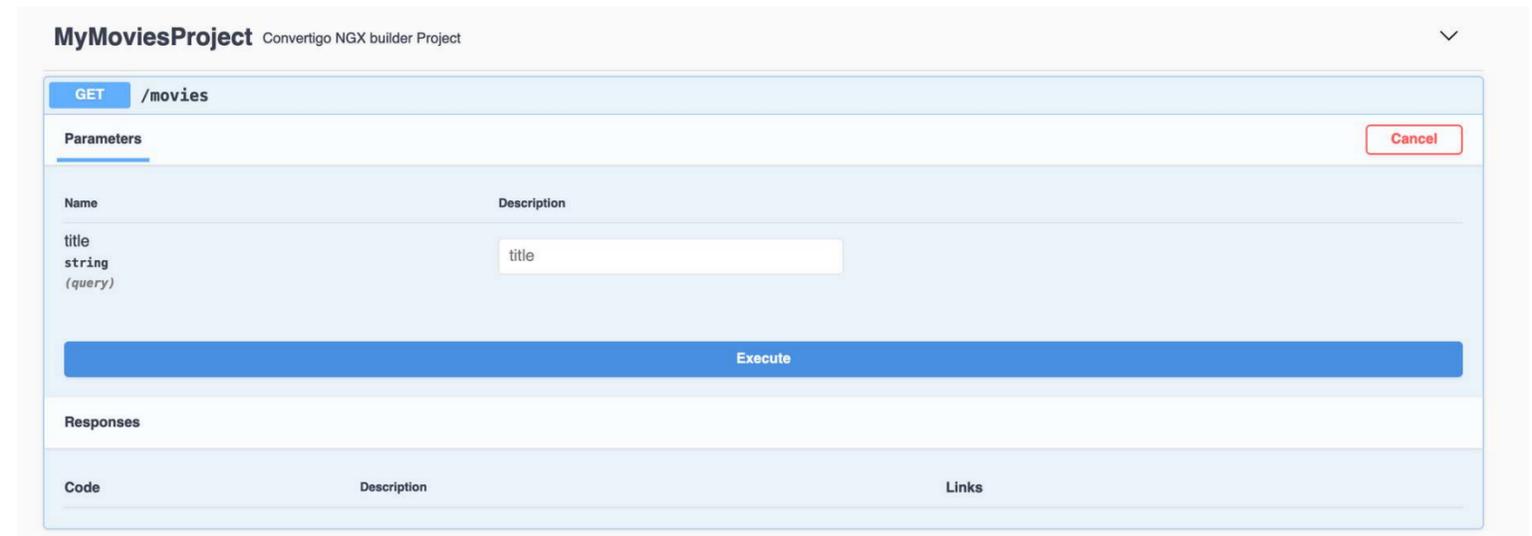


Click on the Try it out button.

Try it out



A **title field** and an **Execute button** appear.



title  
string  
(query)

title

Execute



# 9.4 Test the URL mapper on Swagger

MyMoviesProject Convertigo NGX builder Project

GET /movies

Parameters Cancel

Name	Description
title string (query)	<input type="text" value="title"/>

Execute

Responses

Code	Description	Links
------	-------------	-------



Enter a value in the title field (here "avatar").

Name	Description
title string (query)	<input type="text" value="avatar"/>



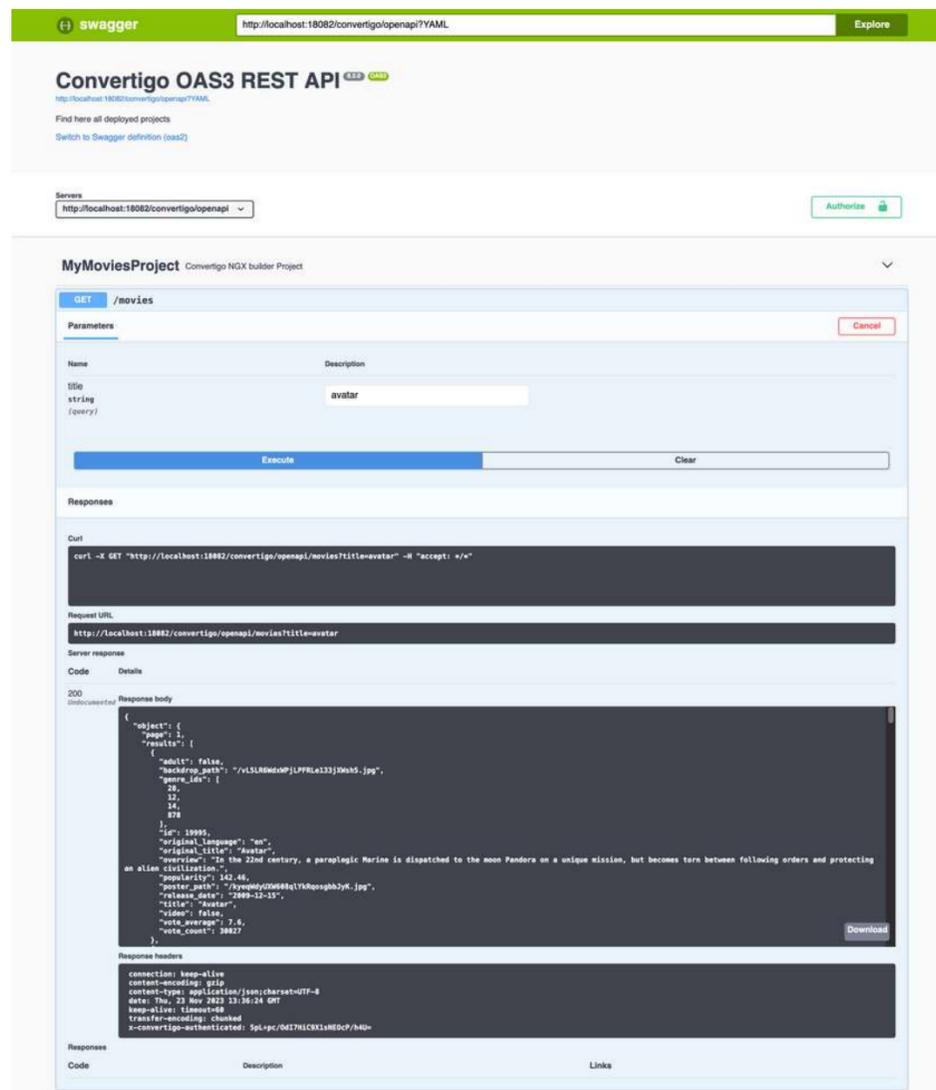
Click on Execute

Execute



# 9.4 Test the URL mapper on Swagger

A response result of the GET /movies request appears in the Swagger



The screenshot shows the Swagger UI interface for the 'Convertigo OAS3 REST API'. The endpoint 'GET /movies' is selected, and the query parameter 'title=avatar' is entered. The 'Execute' button is highlighted. Below, the 'Server response' section shows a 200 status code and a detailed JSON response body for the movie 'Avatar'.



**Request URL**

```
http://localhost:18082/convertigo/openapi/movies?title=avatar
```



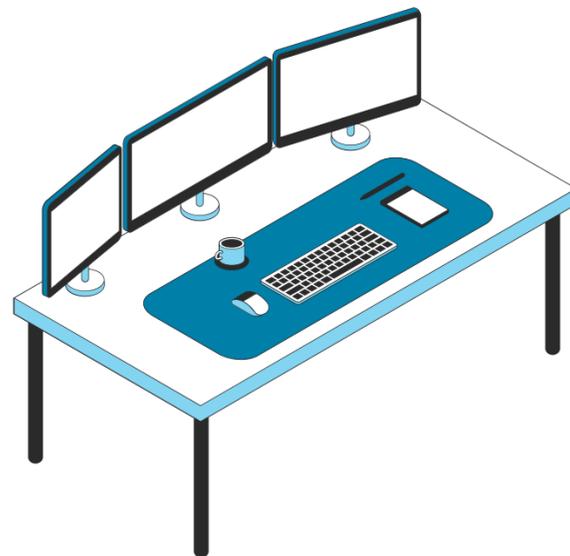
**Server response**

Code	Details
200	<p>Undocumented</p> <p><b>Response body</b></p> <pre>{   "object": {     "page": 1,     "results": [       {         "adult": false,         "backdrop_path": "/vL5LR6WdxWPjLPFRLe133jXWsh5.jpg",         "genre_ids": [           28,           12,           14,           878         ],         "id": 19995,         "original_language": "en",         "original_title": "Avatar",         "overview": "In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization.",         "popularity": 142.46,         "poster_path": "/kyeqWdyUXW608qLYkRqosgbbJyK.jpg",         "release_date": "2009-12-15",         "title": "Avatar",         "video": false,         "vote_average": 7.6,         "vote_count": 30027       }     ]   } }</pre>



# 10 – Nocode database

How to use the NoCode Database.



**10.1** Presenting Baserow

---

**10.2** Set up your Baserow account

---

**10.3** Create a database

---

**10.4** Import CRUD sequences into a project

---

**10.5** Add filters in a table

---

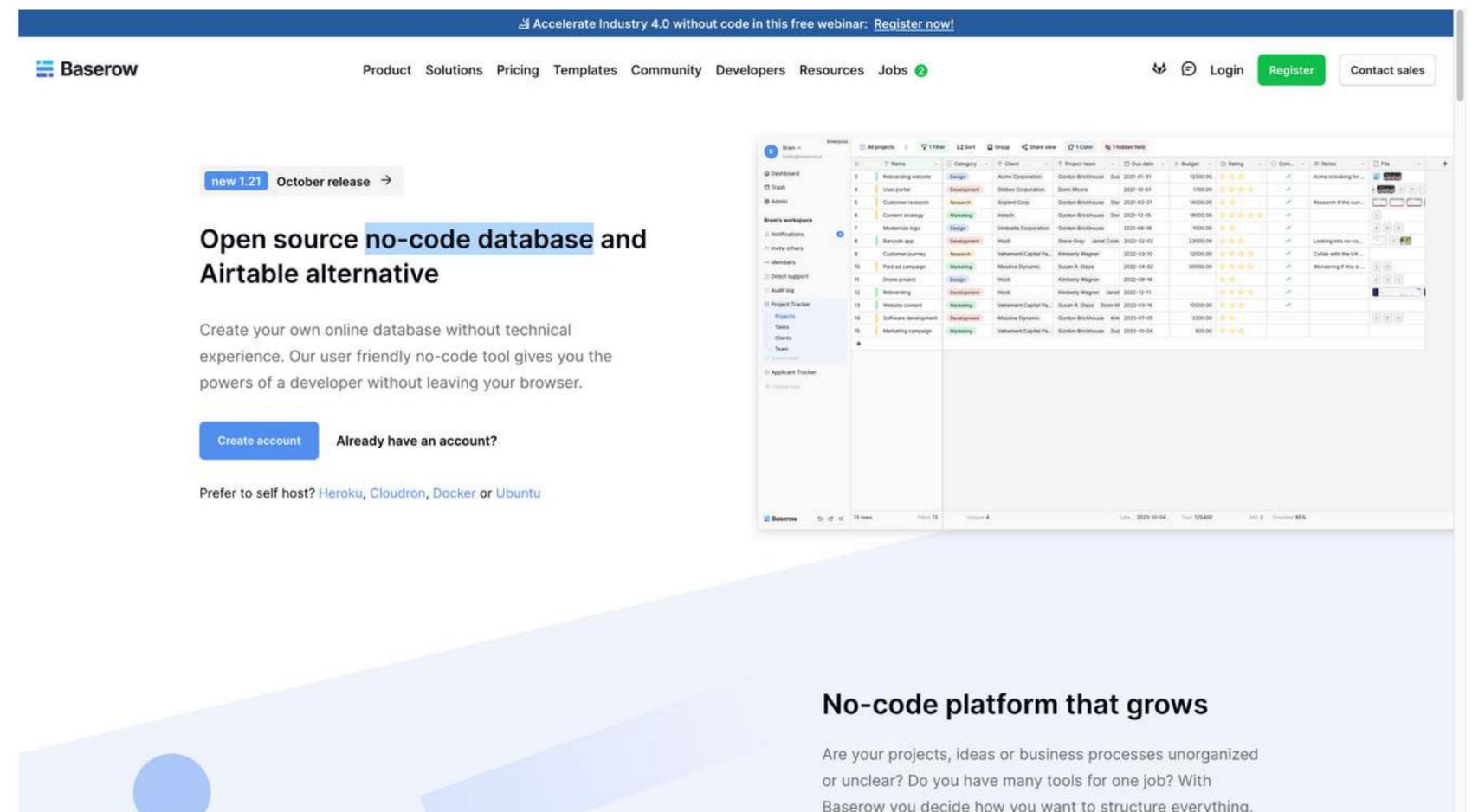
**10.6** Test the CRUD sequences

# 10.1 Presenting Baserow

Convertigo Low code studio integrates Baserow as no-code database.

Baserow is an **open-source no-code database** that **allows users to create databases and web applications without the need for coding.**

It provides a **user-friendly interface** for designing databases, setting up tables, and views for data entry and visualization.



The screenshot shows the Baserow website with a navigation bar and a main content area. The main content area features a headline: "Open source no-code database and Airtable alternative". Below the headline is a sub-headline: "Create your own online database without technical experience. Our user friendly no-code tool gives you the powers of a developer without leaving your browser." There are two buttons: "Create account" and "Already have an account?". Below the buttons is a link: "Prefer to self host? Heroku, Cloudron, Docker or Ubuntu". To the right of the text is a screenshot of a database interface showing a table with columns: Name, Category, Client, Project team, Due date, Budget, Rating, and Comments. The table contains several rows of data.

**No-code platform that grows**

Are your projects, ideas or business processes unorganized or unclear? Do you have many tools for one job? With Baserow you decide how you want to structure everything.

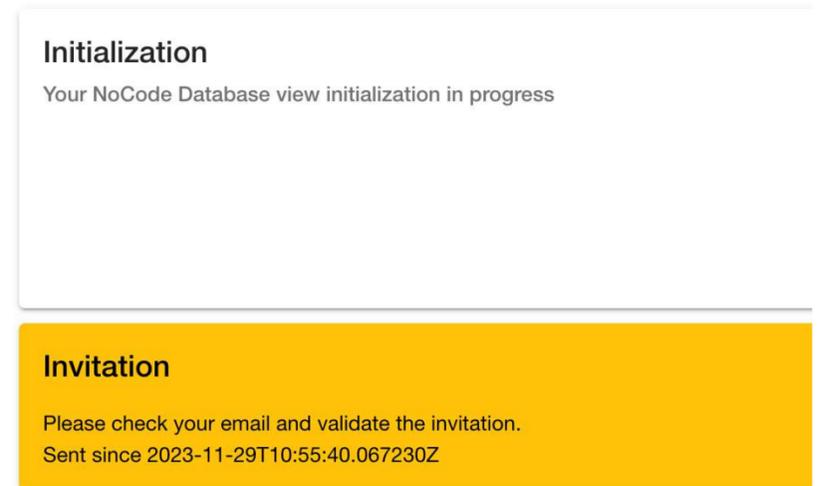
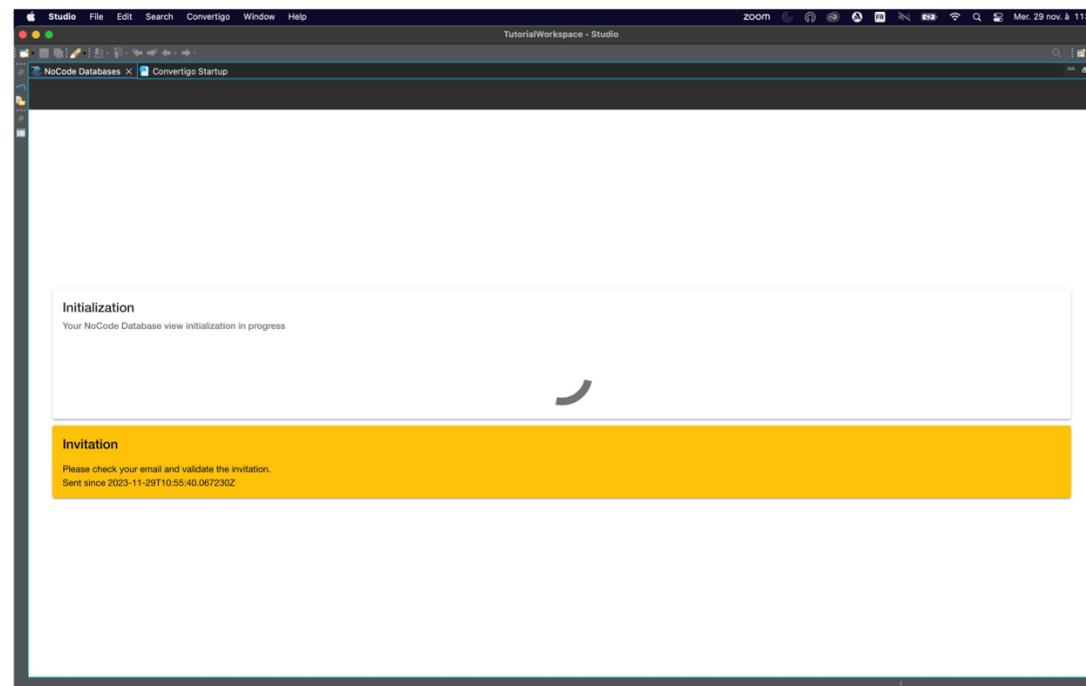
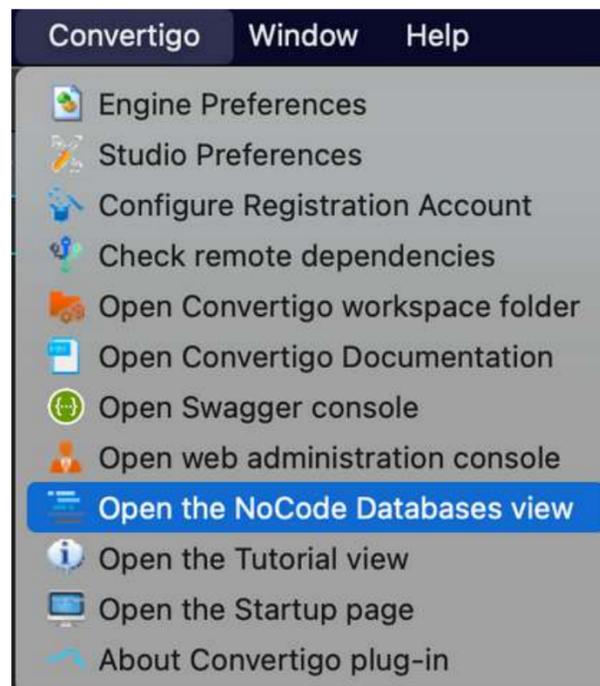


# 10.2 Set up your Baserow account

Let's start by opening the **No Code Database view** in the studio to set up your account.

Click on **Convertigo**, then click on **Open the NoCode Databases view**.

In the **NoCode Databases view**, a message tells you to check your emails to create your profile (The email you used to create your Studio account).

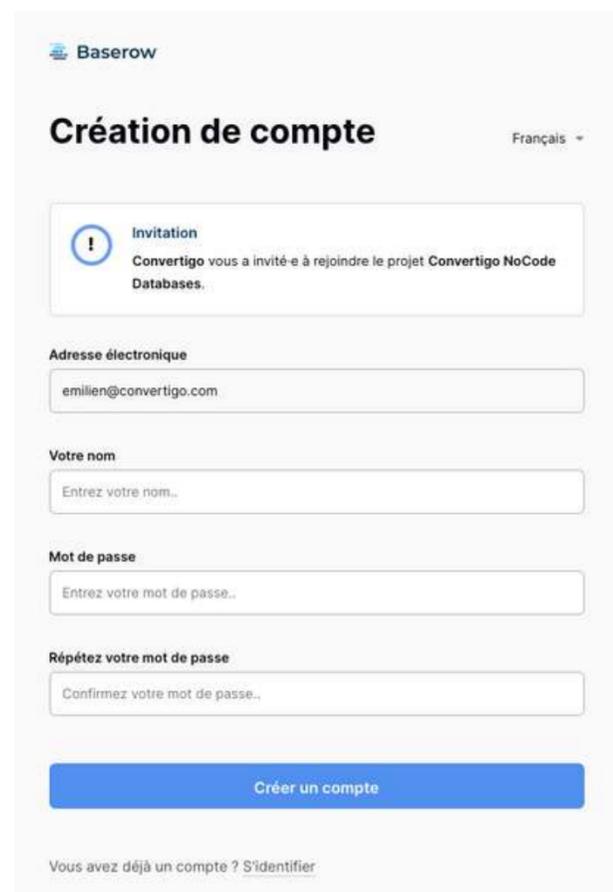


# 10.2 Set up your Baserow account

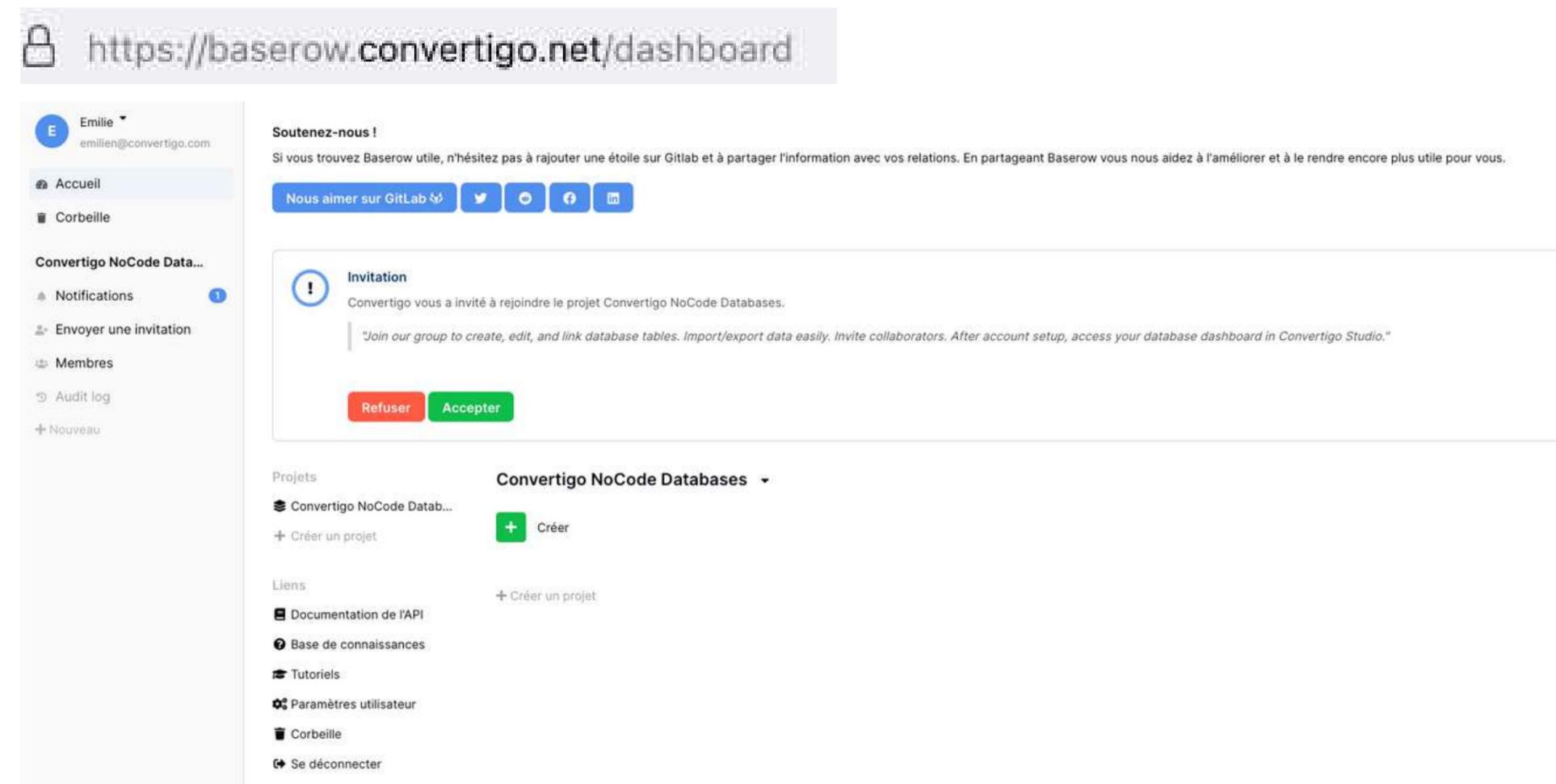
In this email, click on **accept invitation** to open the account creation page.

Fill in the information and Sign up.

Once your account is created, a tab with the **Convertigo baserow dashboard** opens in your browser.



The screenshot shows the Baserow account creation page. At the top left is the Baserow logo. The main heading is "Création de compte" with a language selector set to "Français". Below this is an "Invitation" section with a warning icon and the text: "Convertigo vous a invité-e à rejoindre le projet Convertigo NoCode Databases." The form contains three input fields: "Adresse électronique" (with "emilien@convertigo.com" entered), "Votre nom" (with "Entrez votre nom.." placeholder), and "Mot de passe" (with "Entrez votre mot de passe.." placeholder). A fourth field "Répétez votre mot de passe" (with "Confirmez votre mot de passe.." placeholder) is also present. A blue "Créer un compte" button is at the bottom. A link "Vous avez déjà un compte ? S'identifier" is at the very bottom.



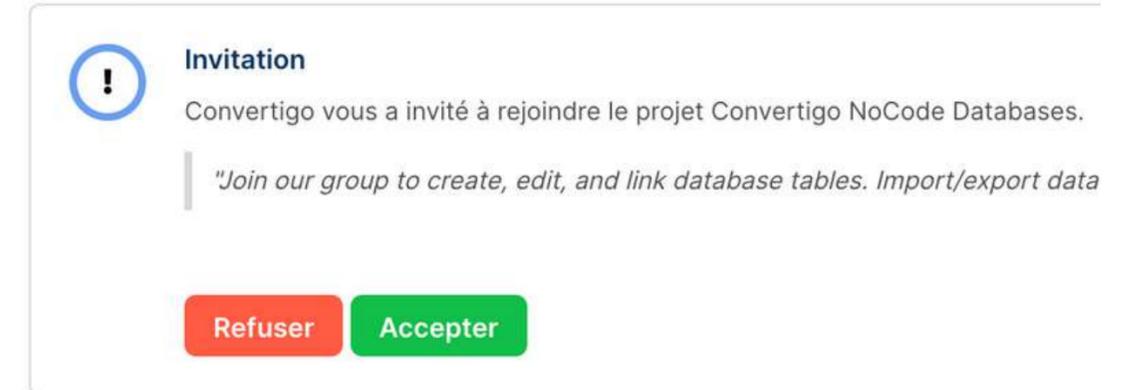
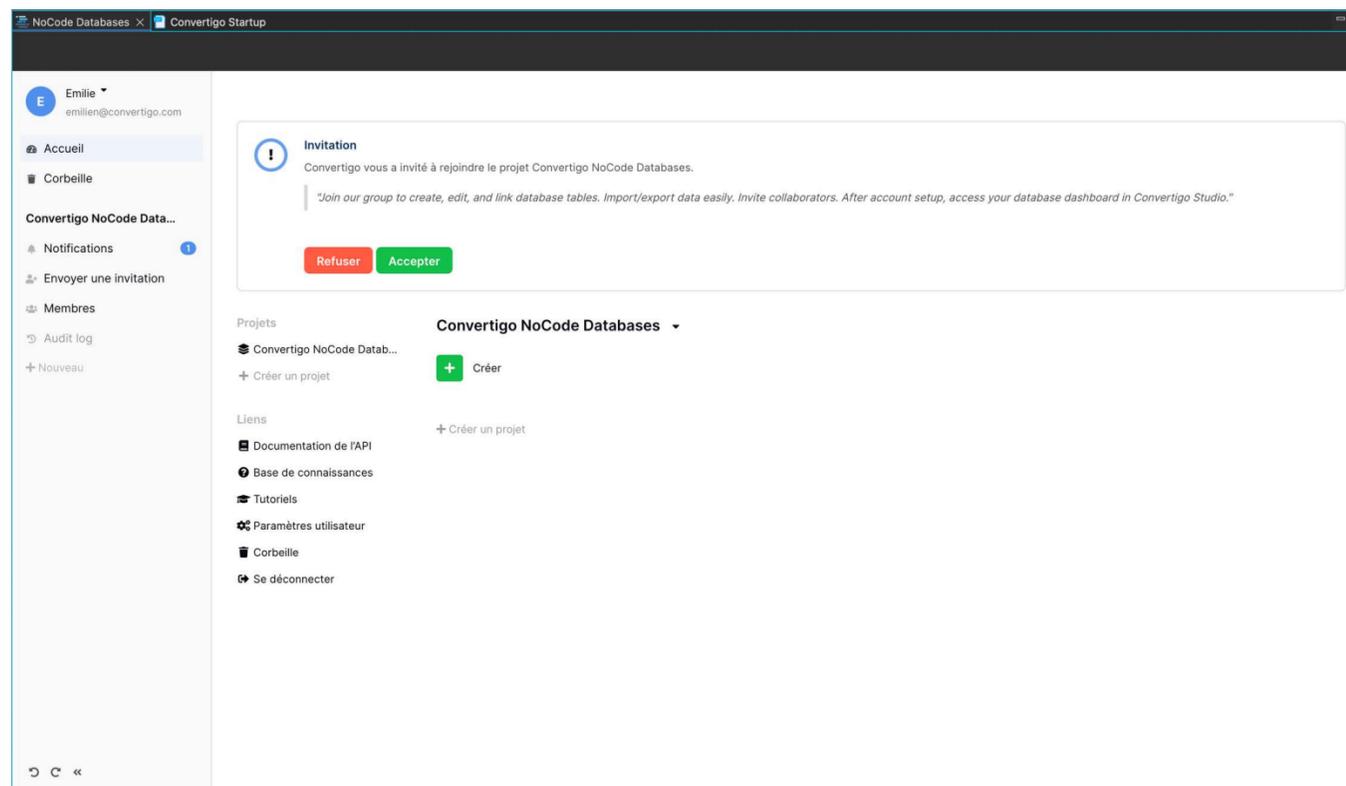
The screenshot shows the Baserow dashboard in a browser. The address bar displays "https://baserow.convertigo.net/dashboard". The user profile "Emilie" (emilien@convertigo.com) is visible in the top left. A navigation sidebar on the left includes "Accueil", "Corbeille", "Convertigo NoCode Data...", "Notifications" (with a badge), "Envoyer une invitation", "Membres", "Audit log", and "Nouveau". The main content area features a "Soutenez-nous !" section with social media links and a "Nous aimer sur GitLab" button. Below this is an "Invitation" section with a warning icon and the text: "Convertigo vous a invité à rejoindre le projet Convertigo NoCode Databases." It includes a quote: "Join our group to create, edit, and link database tables. Import/export data easily. Invite collaborators. After account setup, access your database dashboard in Convertigo Studio." and "Refuser" and "Accepter" buttons. The dashboard also shows a "Projets" section for "Convertigo NoCode Databases" with a "Créer un projet" button, and a "Liens" section with various utility links like "Documentation de l'API", "Base de connaissances", "Tutoriels", "Paramètres utilisateur", "Corbeille", and "Se déconnecter".



# 10.2 Set up your Baserow account

The same dashboard opens in the **NoCode Databases view** of the studio.

Click on **Accept** to join the Convertigo NoCode Databases project (either in the studio or in the browser).



## Projets

Convertigo NoCode Datab...

+ Créer un projet

## Convertigo NoCode Databases ▾

+ Créer

## Liens

Documentation de l'API

Base de connaissances

Tutoriels

Paramètres utilisateur

Corbeille

Se déconnecter

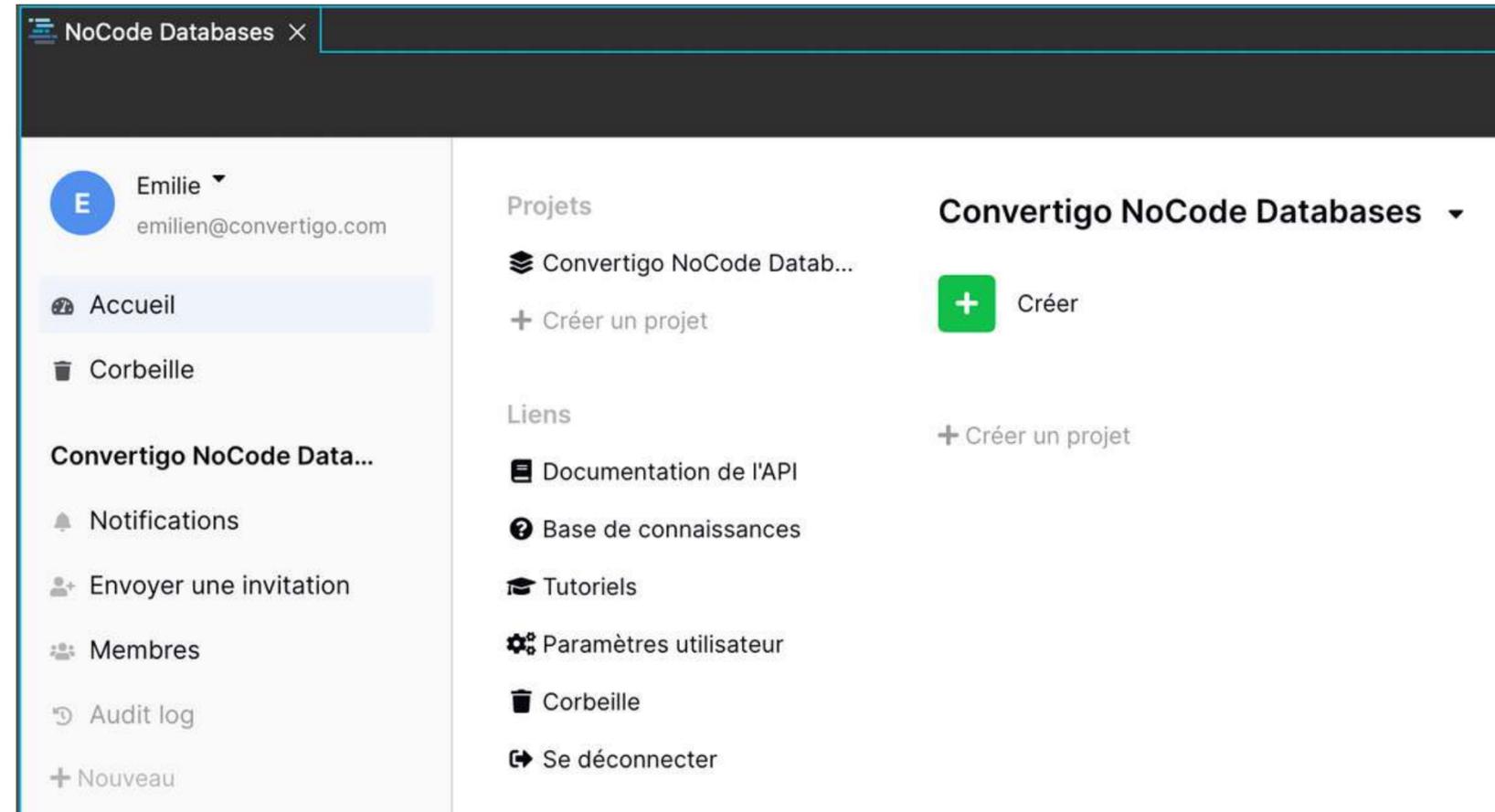
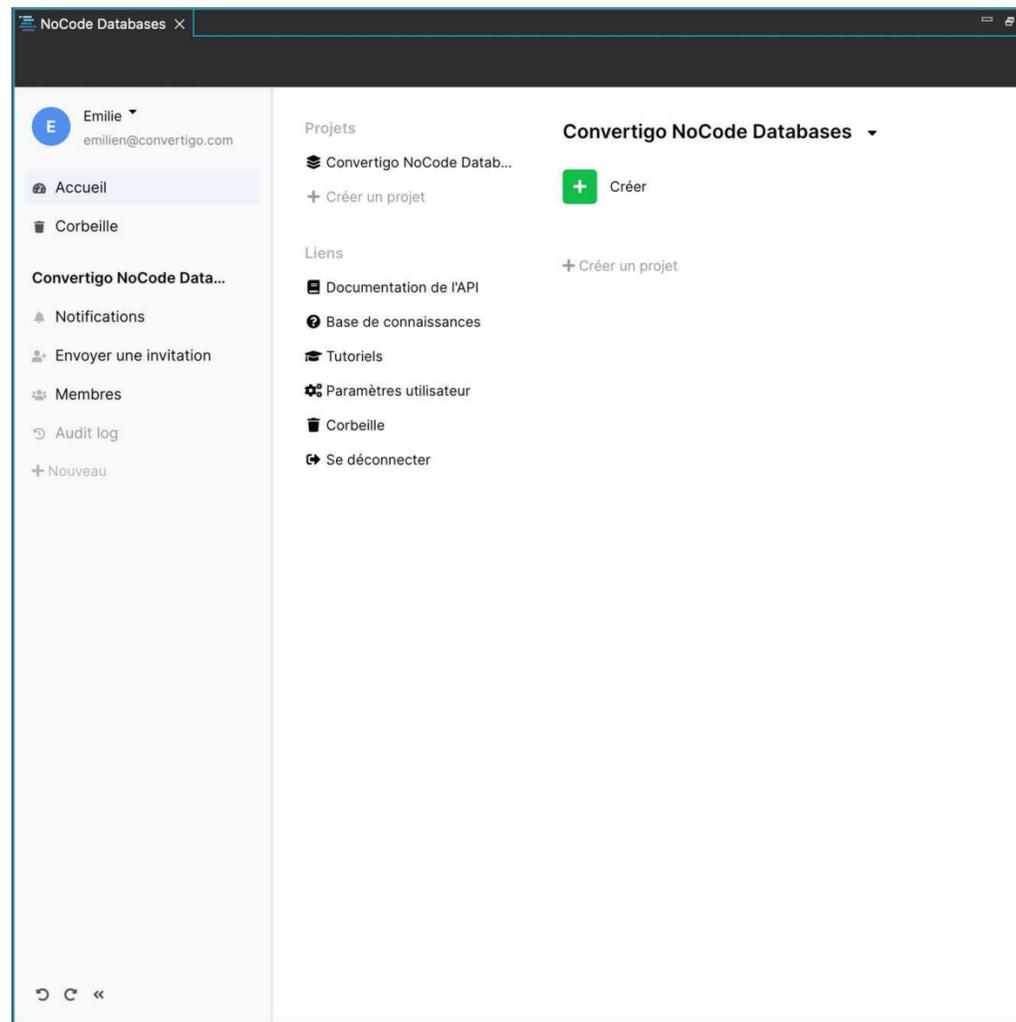
+ Créer un projet



# 10.2 Set up your Baserow account

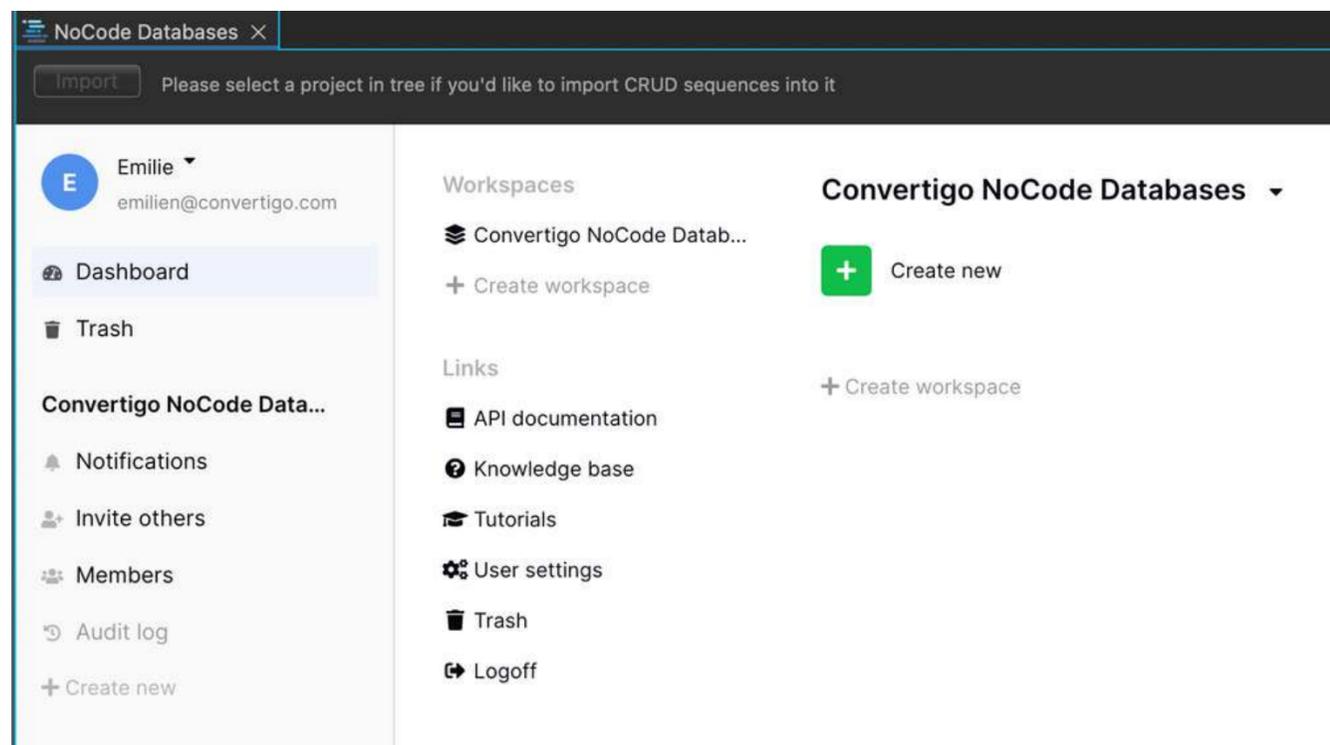


You are now ready to use your nocode database.



# 10.3 Create a database

Let's create a new database from the dashboard in the NoCode Databases view.



Click on **Create new**.

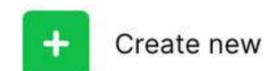
Convertigo NoCode Databases ▾



You can either

- start a new database from scratch
- or select a template

Convertigo NoCode Databases ▾



# 10.3 Create a database

First option :  
you click on **From template**,

Convertigo NoCode Databases ▾

**+ Create new**

- Database
- From template**



You will see a **selection of templates organized in categories**.

Project Tracker Local Business Use this template

Project Tracker All projects Filter Sort 1 hidden field

	A Name	Category	Client	Project lead	Project team	Kickoff date
3	Rebranding website	Design	Acme Corporation	Kimberly Wagner	Gordon Brickhouse Susan R. Glaze	2021-01-01
5	Customer research	Research	Soylent Corp	Janet Cook	Gordon Brickhouse Steve Gray	2021-03-15
7	Modernize logo	Design	Umbrella Corporation		Gordon Brickhouse	2021-05-01
8	Barcode app	Development	Hooli	Gordon Brickhouse	Steve Gray Janet Cook	2021-05-01
4	User portal	Development	Globex Corporation	Kimberly Wagner	Donn Moore	2021-05-01
9	Customer journey	Research	Vehement Capital Pa...		Kimberly Wagner	
6	Content strategy	Marketing	Initech	Kimberly Wagner	Gordon Brickhouse Donn Moore	
10	Paid ad campaign	Marketing	Massive Dynamic		Susan R. Glaze	

8 rows Earliest 2021-01-01



**Templates**

Search templates

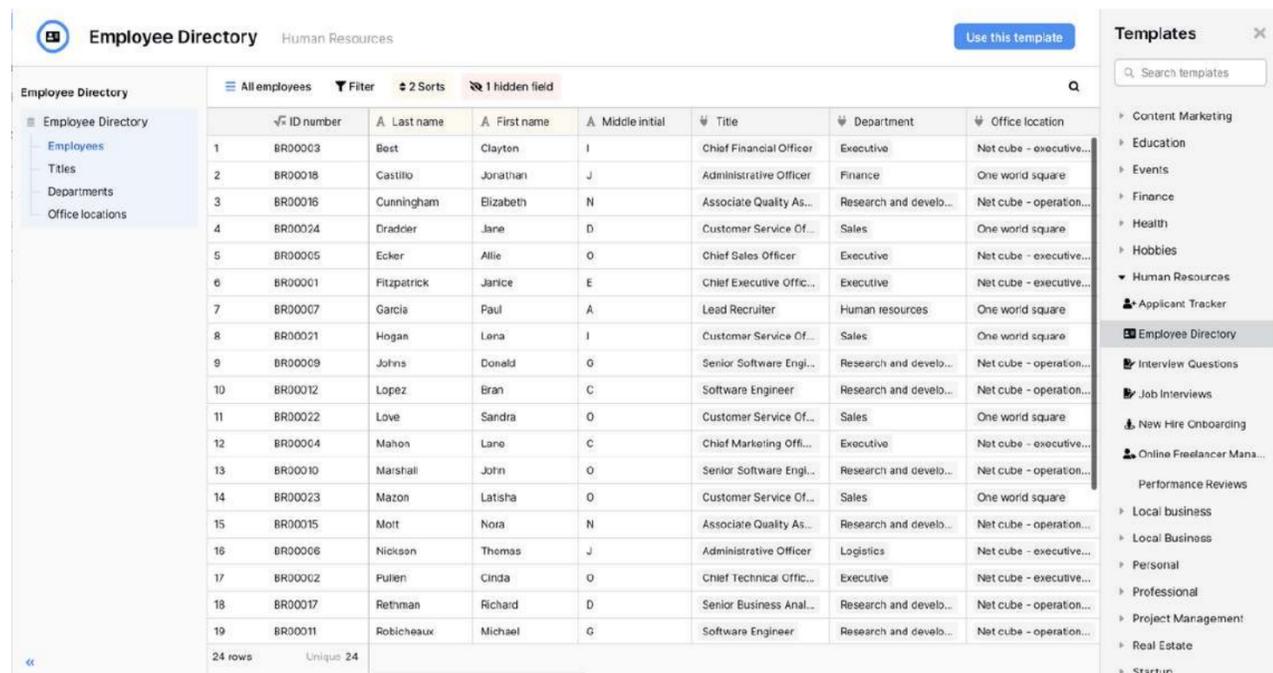
- Content Marketing
- Education
- Events
- Finance
- Health
- Hobbies
- Human Resources
- Local business
- Local Business
- Applicant Tracker
- App Pitch Planner
- Business Expenses
- Business Goal Tracker (...)
- Call Center Log
- Employee Directory
- Furniture, Fixtures, and ...
- Hotel Bookings
- Lightweight CRM
- Meeting Minutes
- Meeting Room Booking



# 10.3 Create a database

Let's select a template to manage an employee list.

Click on **Employee Directory** in the category **Human Resources** to display the **Employee Directory** database.



ID number	Last name	First name	Middle initial	Title	Department	Office location
1	Best	Clayton	I	Chief Financial Officer	Executive	Net cube - executive...
2	Castillo	Jorathian	J	Administrative Officer	Finance	One world square
3	Cunningham	Elizabeth	N	Associate Quality As...	Research and develo...	Net cube - operat...
4	Draddier	Jane	D	Customer Service Of...	Sales	One world square
5	Ecker	Allie	O	Chief Sales Officer	Executive	Net cube - executive...
6	Fitzpatrick	Jarice	E	Chief Executive Offi...	Executive	Net cube - executive...
7	Garcia	Paul	A	Lead Recruiter	Human resources	One world square
8	Hogan	Lena	I	Customer Service Of...	Sales	One world square
9	Johns	Donald	O	Senior Software Engi...	Research and develo...	Net cube - operat...
10	Lopez	Eran	C	Software Engineer	Research and develo...	Net cube - operat...
11	Love	Sandra	O	Customer Service Of...	Sales	One world square
12	Mahon	Lane	C	Chief Marketing Off...	Executive	Net cube - executive...
13	Marshall	John	O	Senior Software Engi...	Research and develo...	Net cube - operat...
14	Mazon	Latisha	O	Customer Service Of...	Sales	One world square
15	Molt	Nota	N	Associate Quality As...	Research and develo...	Net cube - operat...
16	Nickson	Thomas	J	Administrative Officer	Logistics	Net cube - executive...
17	Pullen	Cinda	O	Chief Technical Offi...	Executive	Net cube - executive...
18	Rethman	Richard	D	Senior Business Anal...	Research and develo...	Net cube - operat...
19	Robicheaux	Michael	G	Software Engineer	Research and develo...	Net cube - operat...

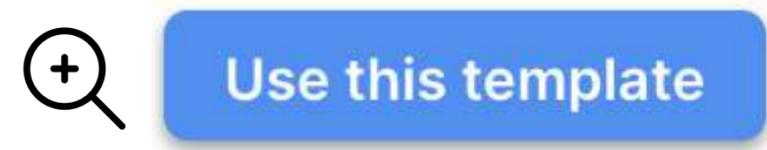


- Human Resources
  - Applicant Tracker
  - Employee Directory**
  - Interview Questions
  - Job Interviews
  - New Hire Onboarding
  - Online Freelancer Mana...
  - Performance Reviews

Employee Directory has 4 tables.

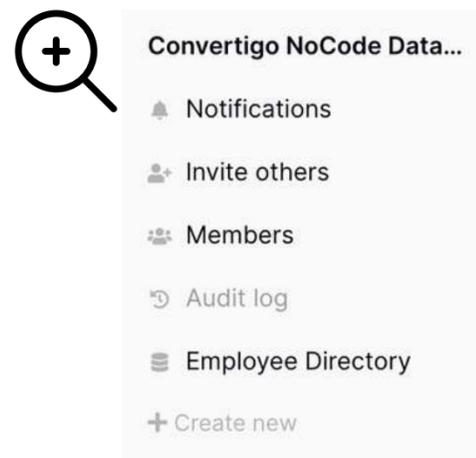
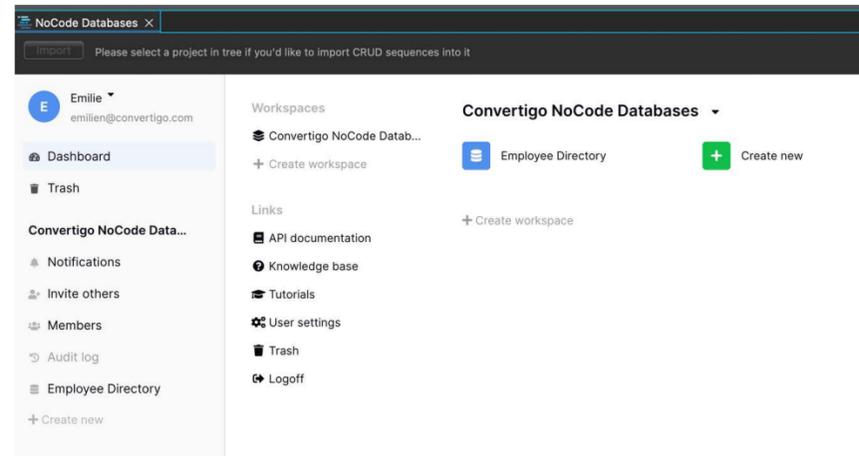


Click on **Use this template.**



# 10.3 Create a database

Once created, the **Employee Directory** database appears in the dashboard of the **NoCode Databases view**.



When you click on **Employee Directory**, you can see the tables and the data of the selected table.

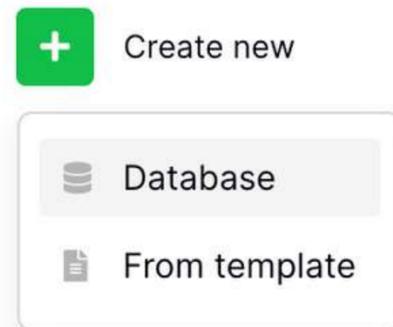
ID	ID number	Last name	First name	Middle initial	Title	Department	Office location	Photo	Local extension number
1	BR00003	Best	Clayton	I	Chief Financial Officer	Executive	Net cube - executive...		x1528
2	BR00018	Castillo	Jonathan	J	Administrative Officer	Finance	One world square		x4050
3	BR00016	Cunningham	Elizabeth	N	Associate Quality As...	Research and develo...	Net cube - operation...		x2084
4	BR00024	Dradder	Jane	D	Customer Service Of...	Sales	One world square		x9899
5	BR00005	Ecker	Allie	O	Chief Sales Officer	Executive	Net cube - executive...		x9008
6	BR00001	Fitzpatrick	Janice	E	Chief Executive Offic...	Executive	Net cube - executive...		x2083
7	BR00007	Garcia	Paul	A	Lead Recruiter	Human resources	One world square		x9834
8	BR00021	Hogan	Lena	I	Customer Service Of...	Sales	One world square		x0720
9	BR00009	Johns	Donald	G	Senior Software Engi...	Research and develo...	Net cube - operation...		x6300
10	BR00012	Lopez	Bran	C	Software Engineer	Research and develo...	Net cube - operation...		x4802
11	BR00022	Love	Sandra	O	Customer Service Of...	Sales	One world square		x1950
12	BR00004	Mahon	Lane	C	Chief Marketing Offi...	Executive	Net cube - executive...		x8338
13	BR00010	Marshall	John	O	Senior Software Engi...	Research and develo...	Net cube - operation...		x6302
14	BR00023	Mazon	Latisha	O	Customer Service Of...	Sales	One world square		x5470
15	BR00015	Mott	Nora	N	Associate Quality As...	Research and develo...	Net cube - operation...		x0314
16	BR00006	Nickson	Thomas	J	Administrative Officer	Logistics	Net cube - executive...		x5557
17	BR00002	Pullen	Cinda	O	Chief Technical Offic...	Executive	Net cube - executive...		x8170
18	BR00017	Rethman	Richard	D	Senior Business Anal...	Research and develo...	Net cube - operation...		x3210
19	BR00011	Robicheaux	Michael	G	Software Engineer	Research and develo...	Net cube - operation...		x1942
20	BR00019	Rothman	Eric	A	Administrative Officer	Finance	One world square		x2313
21	BR00008	Tran	Rodney	E	Assistant Recruiter	Human resources	One world square		x5049
22	BR00020	Tripiett	Judith	O	Administrative Officer	Marketing	Net cube - operation...		x4750

ID	ID number	Last name	First name	Middle initial	Title	Department	Office location	Photo	Local extension number
1	BR00003	Best	Clayton	I	Chief Financial Officer	Executive	Net cube - executive...		x1528
2	BR00018	Castillo	Jonathan	J	Administrative Officer	Finance	One world square		x4050
3	BR00016	Cunningham	Elizabeth	N	Associate Quality As...	Research and develo...	Net cube - operation...		x2084
4	BR00024	Dradder	Jane	D	Customer Service Of...	Sales	One world square		x9899
5	BR00005	Ecker	Allie	O	Chief Sales Officer	Executive	Net cube - executive...		x9008
6	BR00001	Fitzpatrick	Janice	E	Chief Executive Offic...	Executive	Net cube - executive...		x2083
7	BR00007	Garcia	Paul	A	Lead Recruiter	Human resources	One world square		x9834
8	BR00021	Hogan	Lena	I	Customer Service Of...	Sales	One world square		x0720
9	BR00009	Johns	Donald	G	Senior Software Engi...	Research and develo...	Net cube - operation...		x6300
10	BR00012	Lopez	Bran	C	Software Engineer	Research and develo...	Net cube - operation...		x4802
11	BR00022	Love	Sandra	O	Customer Service Of...	Sales	One world square		x1950
12	BR00004	Mahon	Lane	C	Chief Marketing Offi...	Executive	Net cube - executive...		x8338
13	BR00010	Marshall	John	O	Senior Software Engi...	Research and develo...	Net cube - operation...		x6302

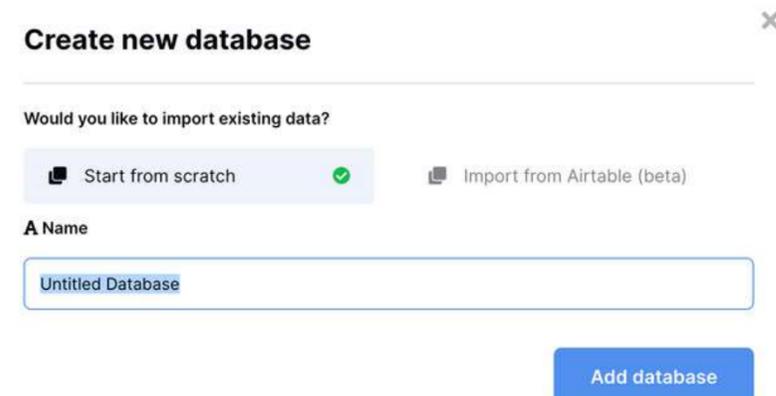


# 10.3 Create a database

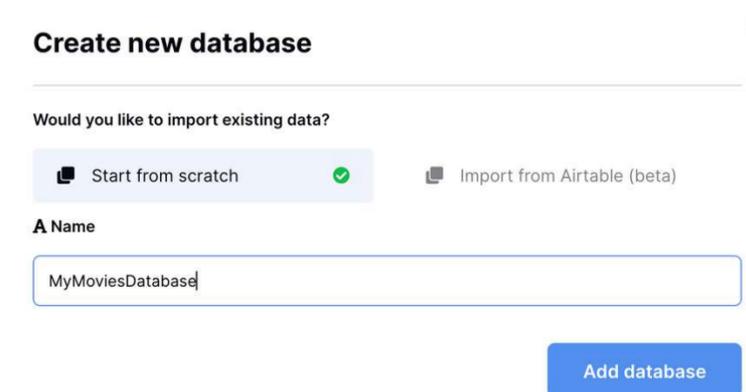
Second option:  
you click on **Database**



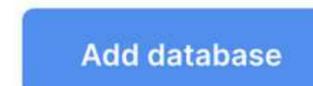
The **Create new database window** appears.



Choose a name for the **new database**.

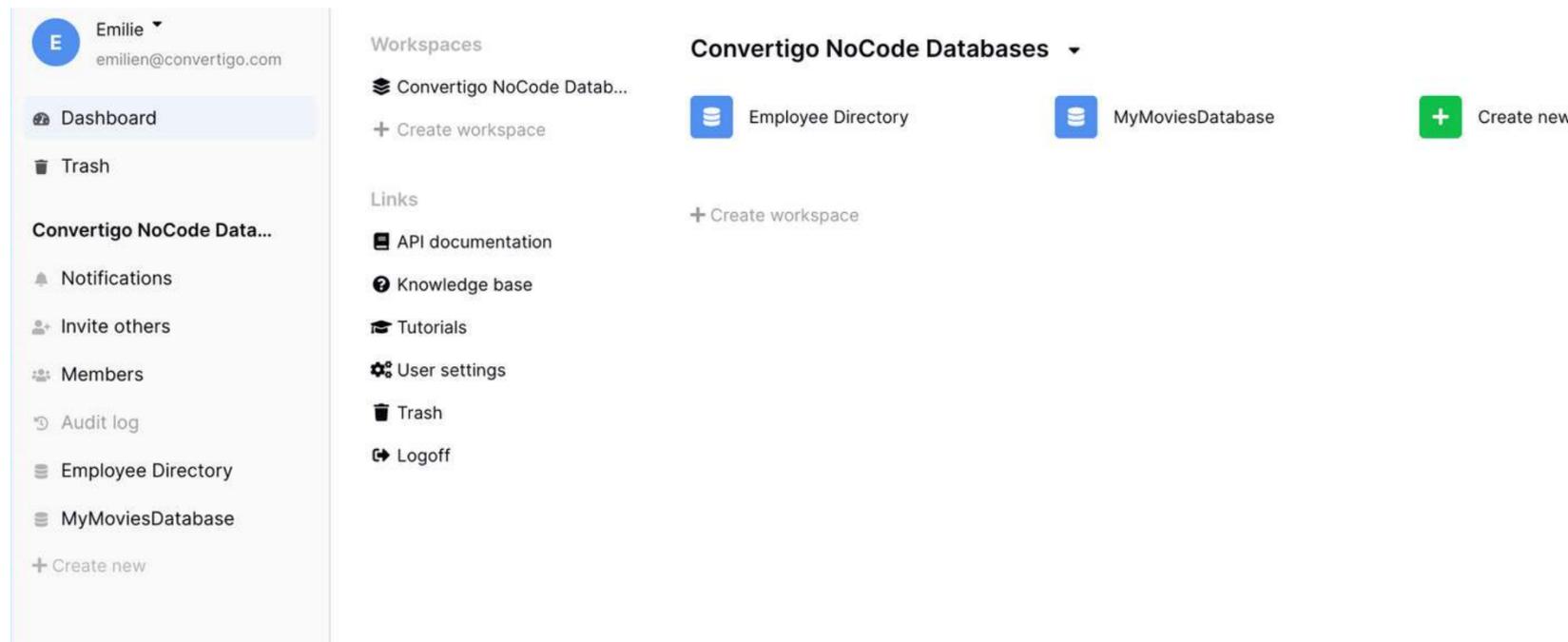


Click on **Add database**.

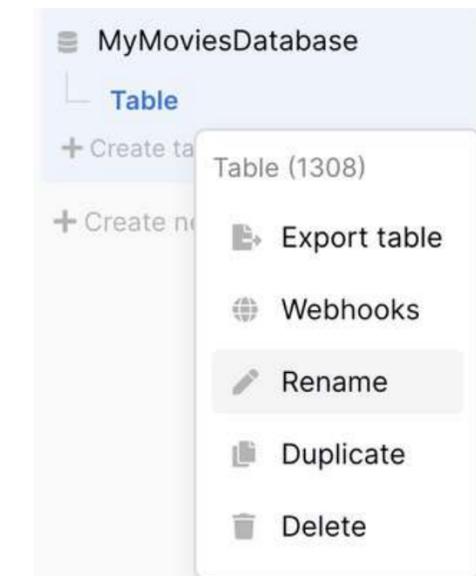


# 10.3 Create a database

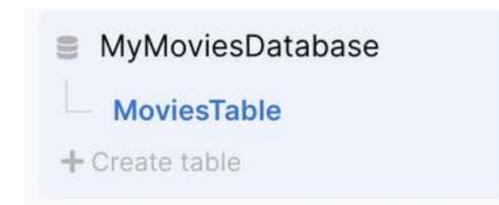
Once created,  
the **MyMoviesDatabase** appears  
in the dashboard of the **NoCode Databases view**.



**MyMoviesDatabase**  
has a table by default



Rename it as **MoviesTable**



# 10.3 Create a database

Click on MoviesTable to display it.

Convertigo NoCode Data...

- Notifications
- Invite others
- Members
- Audit log
- Employee Directory
- MyMoviesDatabase
  - MoviesTable**
  - + Create table
- + Create new



MoviesTable has a few fields by default.

Emilie emilien@convertigo.com

Grid Filter Sort Share view Color Hide Fields

	A Name	Notes	Active
1			
2			
+			



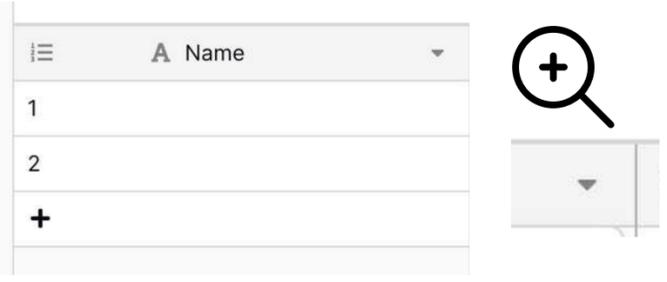
	A Name	Notes	Active
1			
2			
+			



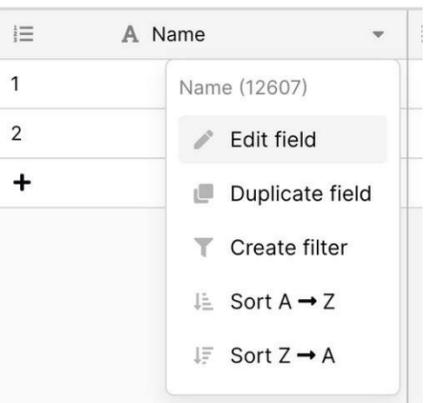
# 10.3 Create a database

Let's create a "Title" field in MoviesTable by editing the Name field.

Select **Single line text** as type of data for **Title field**.



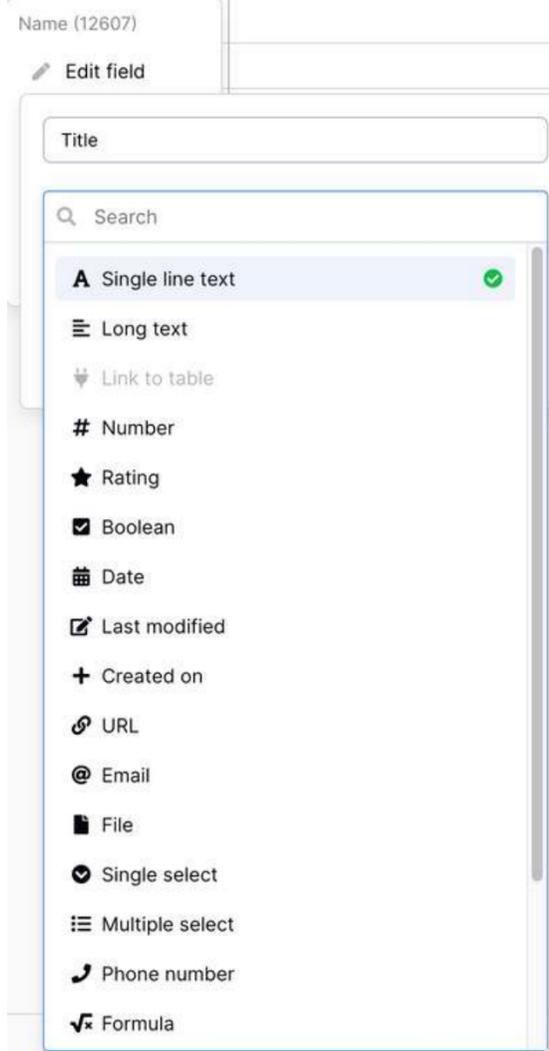
Click on the down arrow to display the menu



The context menu includes options: Edit field, Duplicate field, Create filter, Sort A → Z, and Sort Z → A.



Change Name to Title



The search menu lists various data types: Single line text (selected), Long text, Link to table, Number, Rating, Boolean, Date, Last modified, Created on, URL, Email, File, Single select, Multiple select, Phone number, and Formula.



# 10.3 Create a database

Now we have a **Title** field in **MoviesTable**

A Title
1
2
+

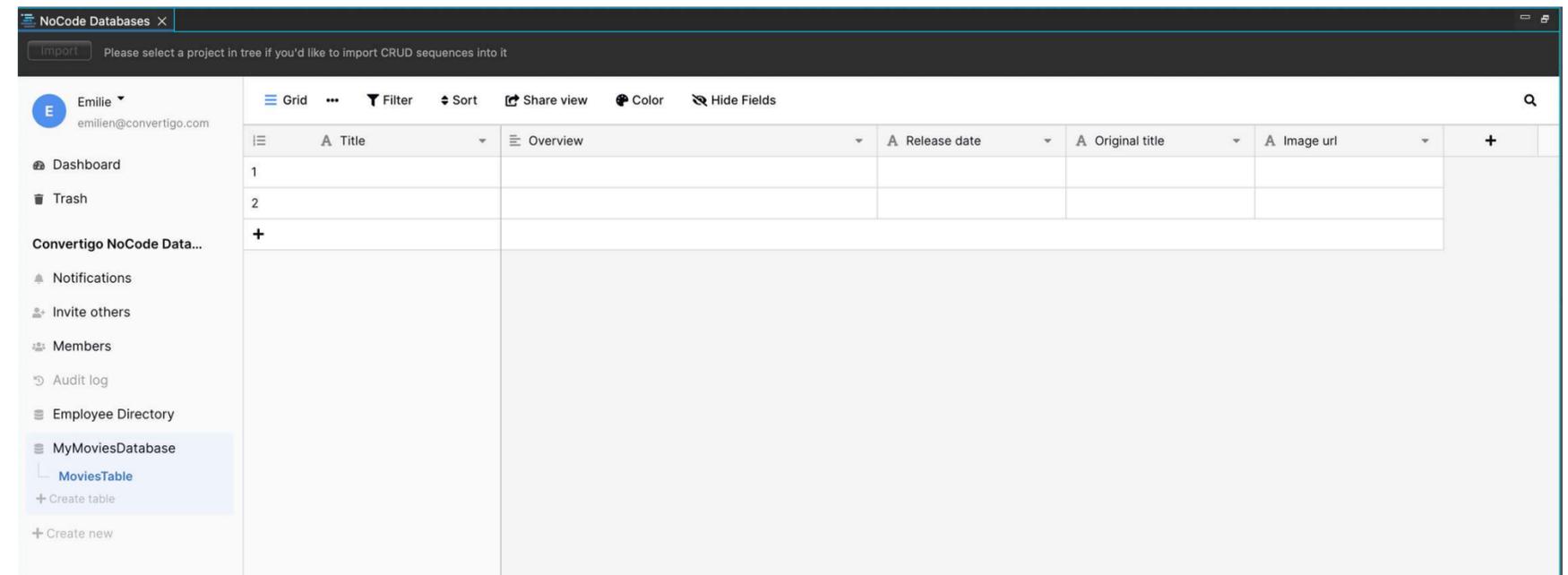


Let's do the same thing and add a few more new fields in **MoviesTable**.



Now we have 5 fields in **MoviesTable**:

**Title, Overview, Release Date, Original title and Image url.**



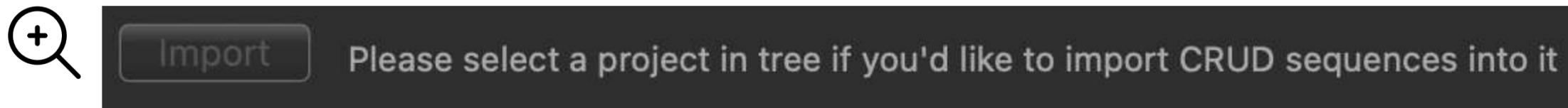
# 10.4 Import CRUD sequences into a project

You are now ready to use your nocode database.



The screenshot shows the 'NoCode Databases' interface. At the top, there is a dark grey bar with an 'Import' button and the text 'Please select a project in tree if you'd like to import CRUD sequences into it'. Below this, the user profile 'Emilie' (emilien@convertigo.com) is visible on the left. The main area shows a table with columns: 'Title', 'Overview', 'Release date', 'Original title', and 'Image url'. The table has three rows, with the first two containing the numbers '1' and '2', and the third containing a '+' sign. Above the table, there are controls for 'Grid', 'Filter', 'Sort', 'Share view', 'Color', and 'Hide Fields'.

When you select a project and click on **Import**,  
**CRUD sequences** will automatically be **created from baserow definitions of the table you selected**,  
then **imported in your project**.

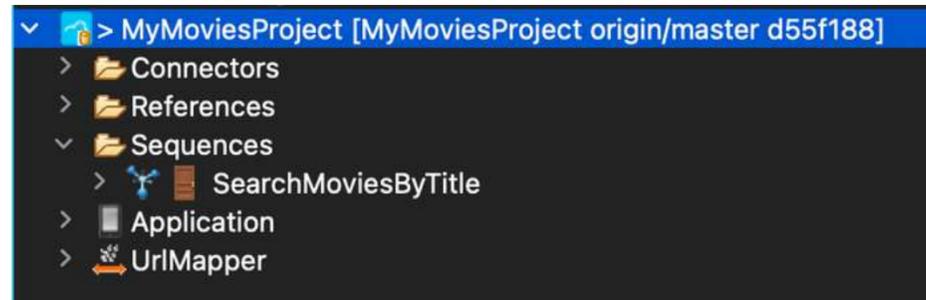


A close-up of the 'Import' button and the text 'Please select a project in tree if you'd like to import CRUD sequences into it'. A magnifying glass icon is positioned to the left of the button.

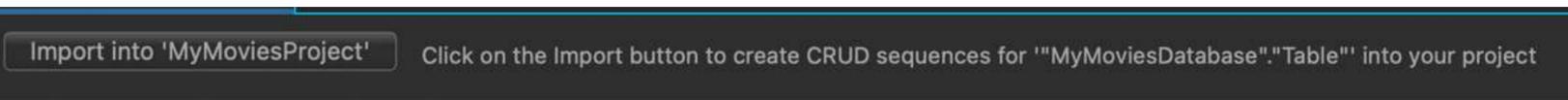
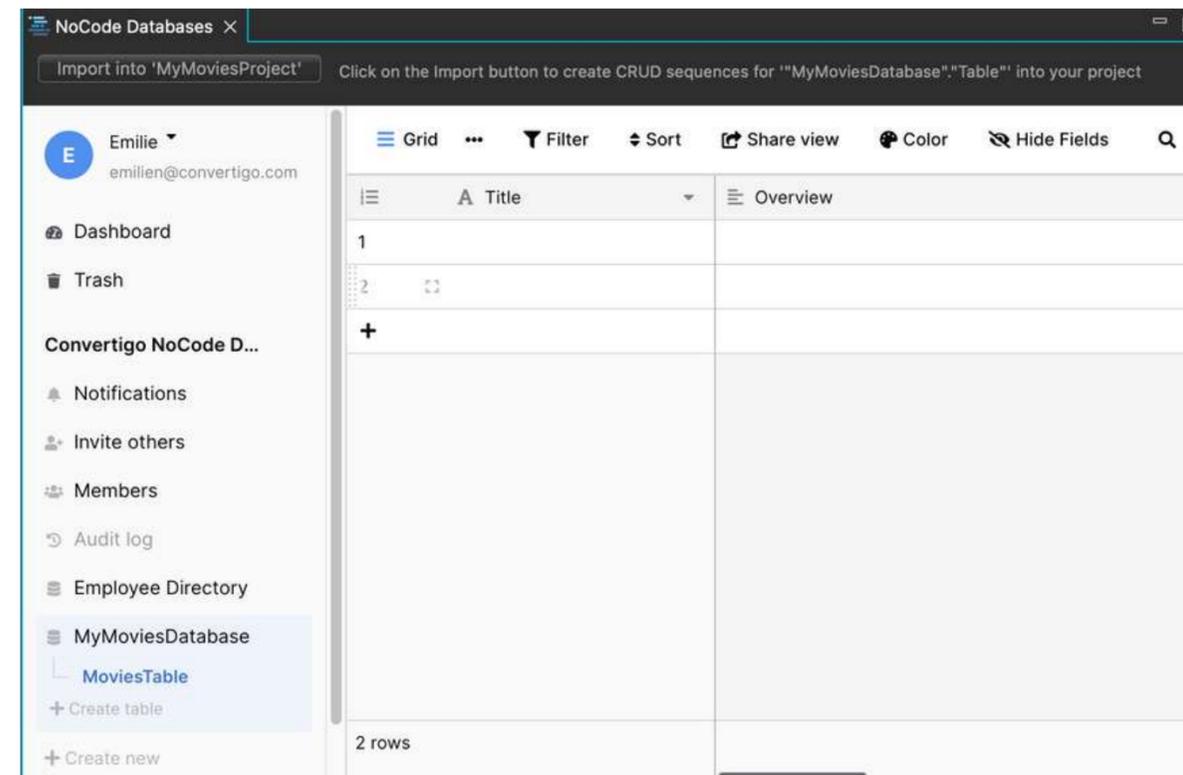


# 10.4 Import CRUD sequences into a project

In the Project view,  
click on MyMoviesProject to select it.

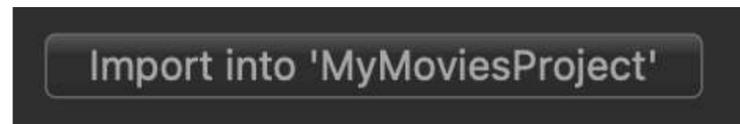


In the NoCode Databases view,  
the Import button text has changed  
from **Import** to **Import into "MyMoviesProject"**.

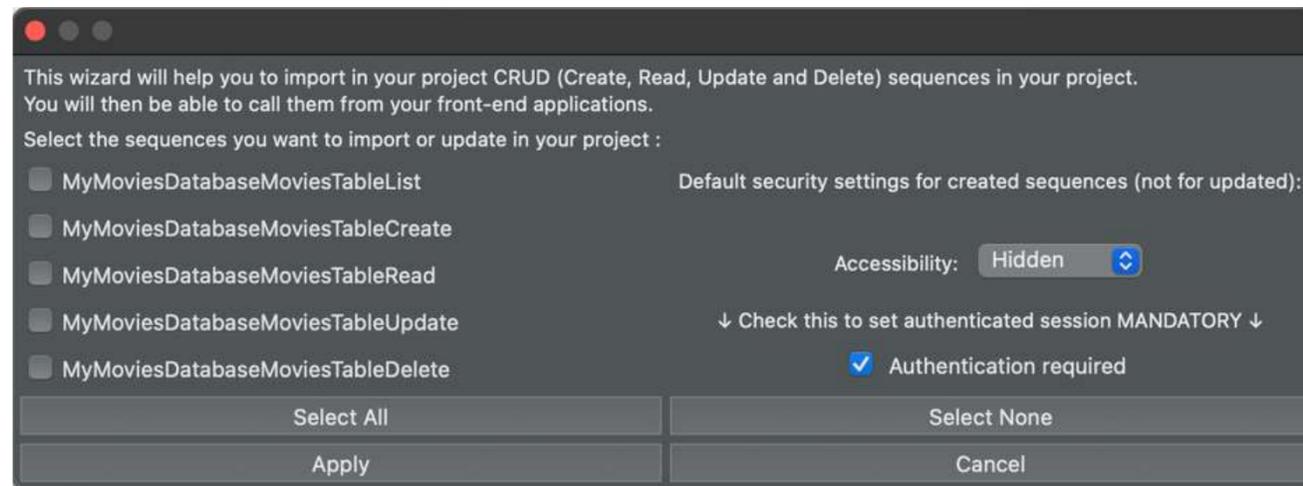


# 10.4 Import CRUD sequences into a project

In the NoCode Databases view,  
click on **Import into "MyMoviesProject"**.



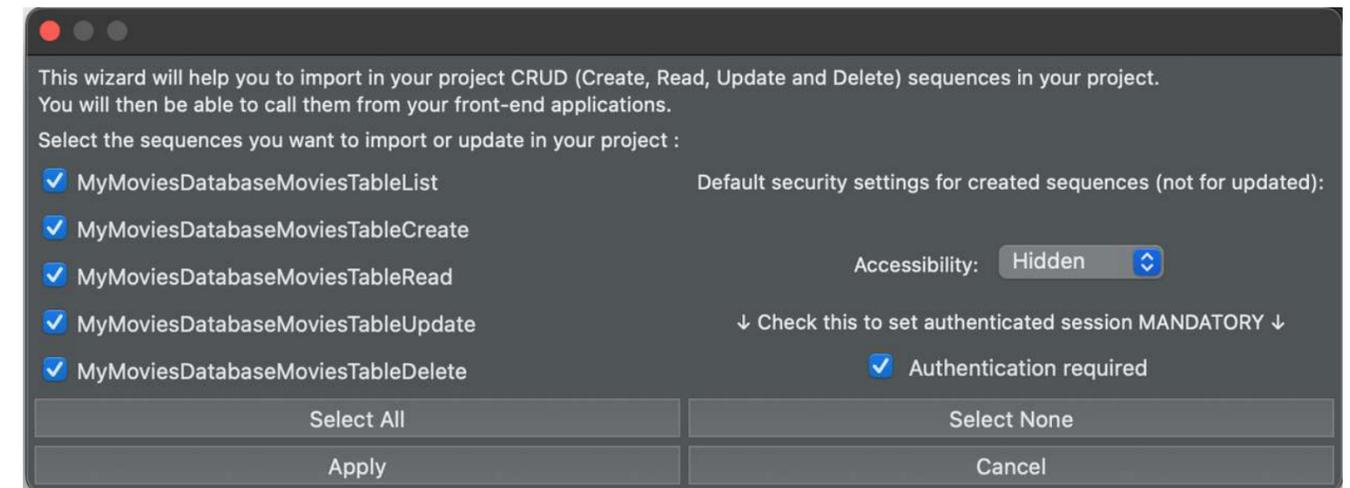
A window allowing you  
to automatically create and import CRUD sequences  
into your project appears.



Click on **Select All**  
to select all CRUD sequences .



Select All



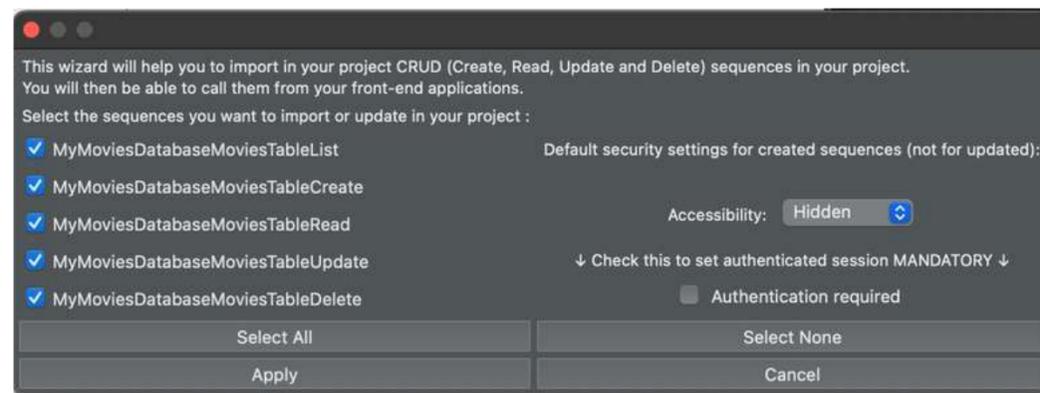
# 10.4 Import CRUD sequences into a project

We don't use authentication  
so **uncheck Authentication required.**



↓ Check this to set authenticated session MANDATORY ↓

Authentication required



↓ Check this to set authenticated session MANDATORY ↓

Authentication required



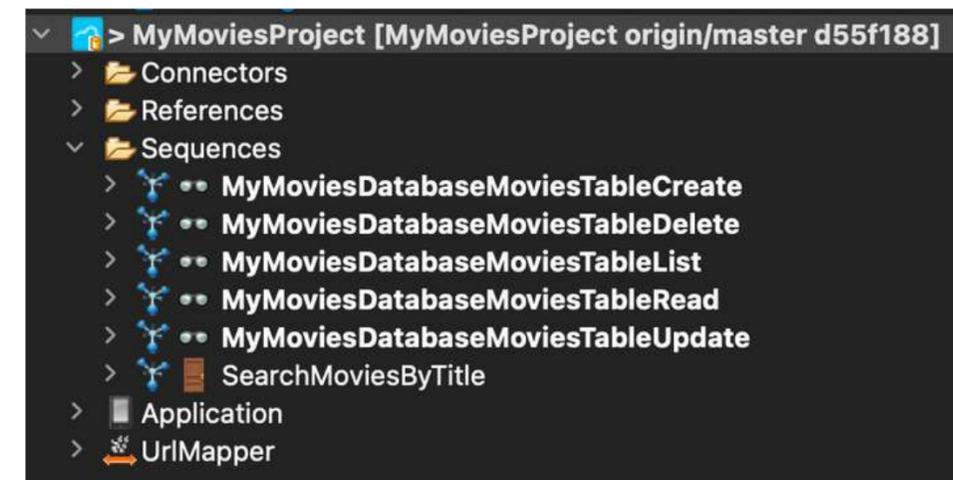
Click on Apply



Apply



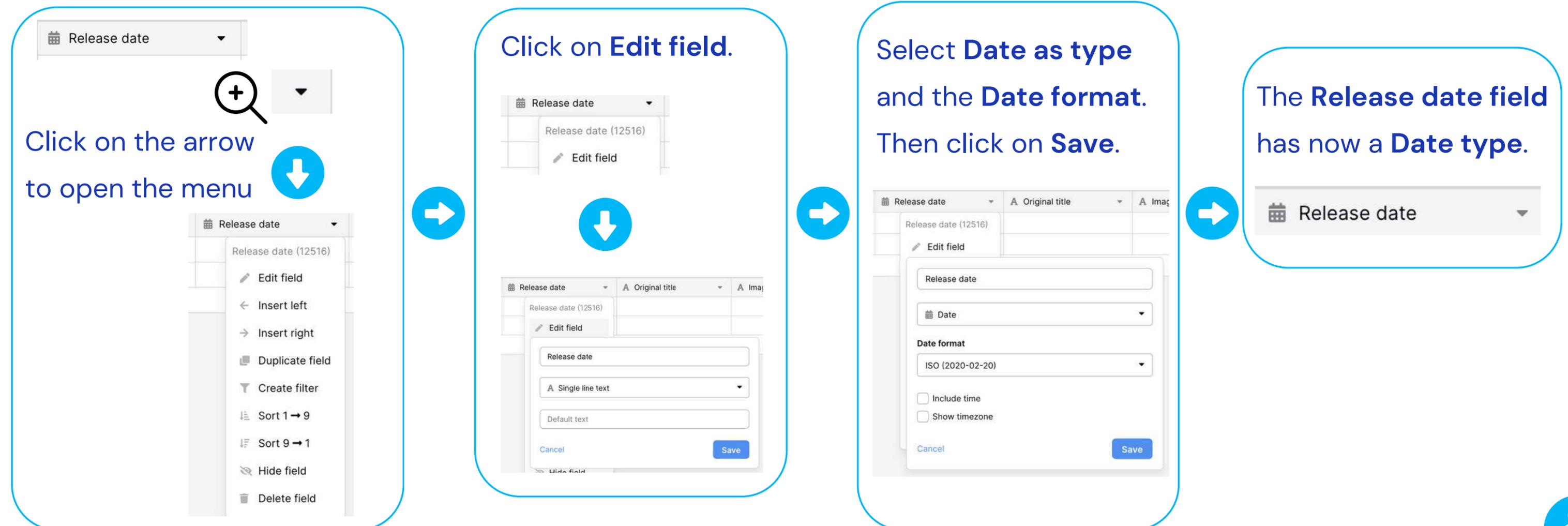
The CRUD sequences were imported  
and appear in **MyMoviesProject** in **Project view.**



# 10.4 Import CRUD sequences into a project

If you change anything in the table, the **table definition in Baserow changes** and you have to **reimport the table**.

For example, let's **edit the Release date field** to change the **type** from Single line text to **Date**.

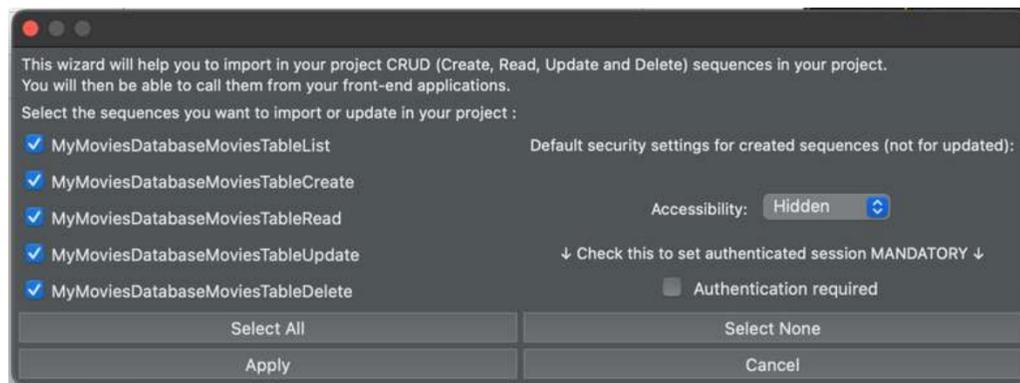


# 10.4 Import CRUD sequences into a project

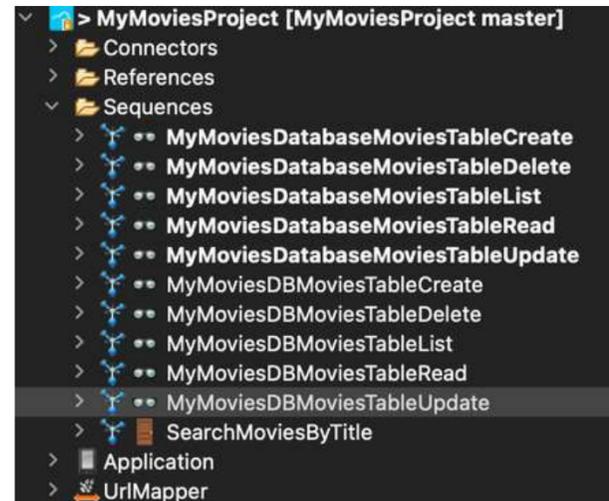
In the NoCode Databases view, click on **Import into "MyMoviesProject"**.



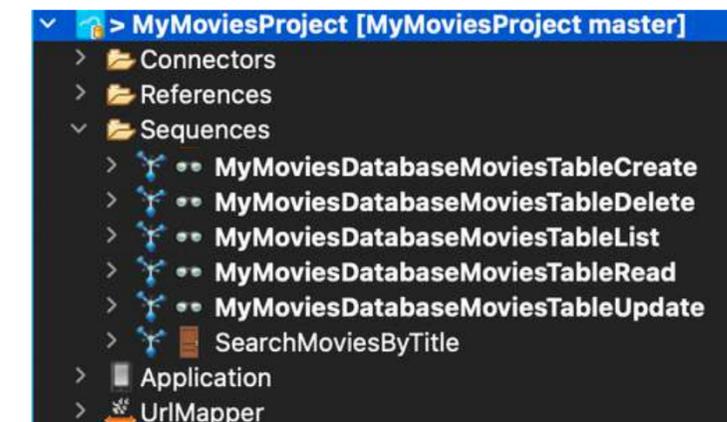
Click on **Select All** then click on **Apply**.



The new CRUD sequences were imported and appear in **MyMoviesProject** in **Project view**.

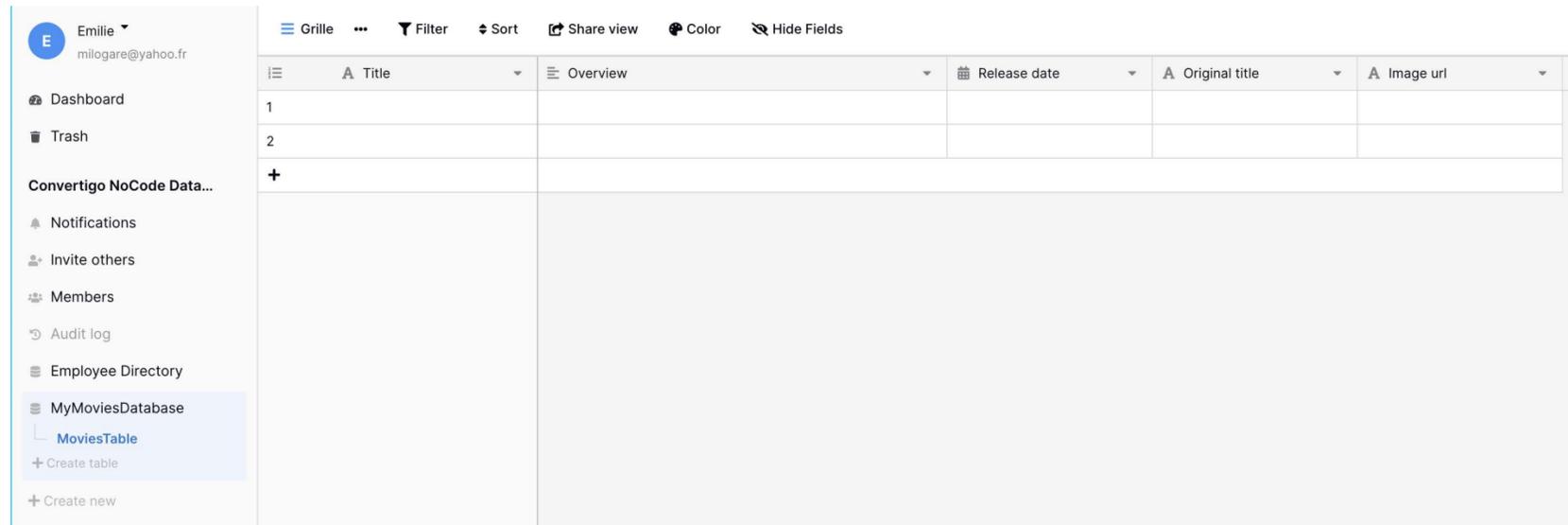


Delete the old CRUD sequences.



# 10.5 Add filters in a table

Let's say you want to **add a filter** on the **Release Date** field to **select only the movies released after a specific date**.





**Grille**

 We are in the **Grille view**.  
 To **add filters**, we need to **add a new view**.

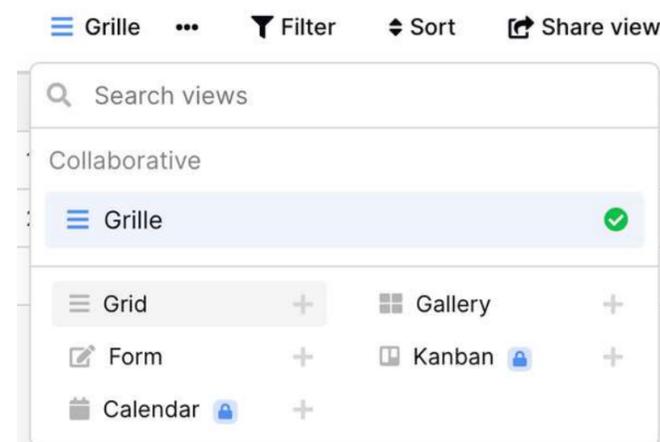


# 10.5 Add filters in a table



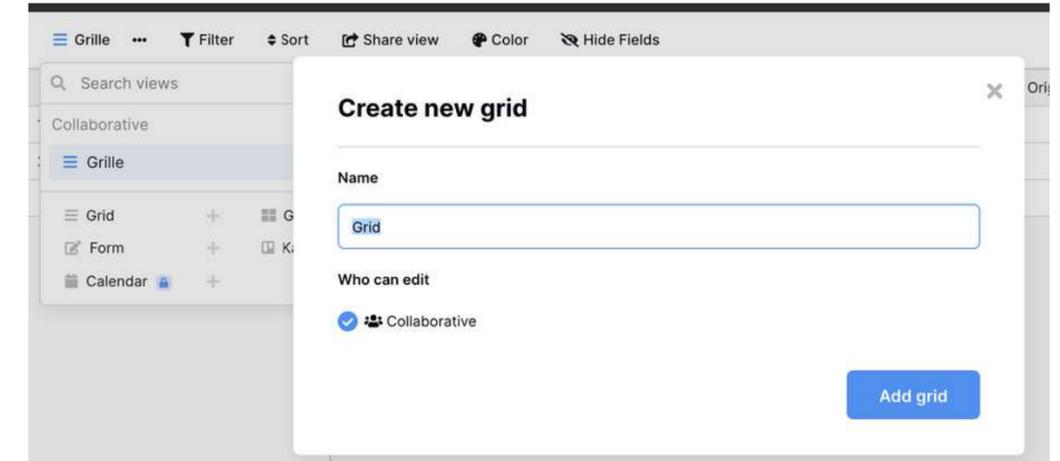
	A Title	Overview	Release date	A Original title	A Image url
1					
2					

  **Grille** Click on the **view name** (Grille, in our case) to **open the views menu**.

Search views	
Collaborative	
 Grille	<input checked="" type="checkbox"/>
 Grid	<input type="checkbox"/>
 Form	<input type="checkbox"/>
 Calendar	<input type="checkbox"/>
 Gallery	<input type="checkbox"/>
 Kanban	<input type="checkbox"/>

  **Grid** Click on **Grid** to open the **Create new grid window**.

**Create new grid**

Name:

Who can edit:  Collaborative



Rename the **Grid** as **WithFilter** and click on **Add grid**.



**Create new grid**

Name:

Who can edit:  Collaborative



# 10.5 Add filters in a table

We are now in the new view **WithFilter**.

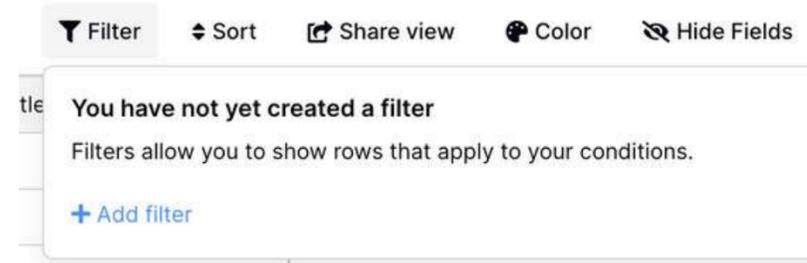
	Title	Overview	Release date	Original title	Image url
1					
2					
+					



 **Filter** Click on **Filter**.



Click on **Add filter**.



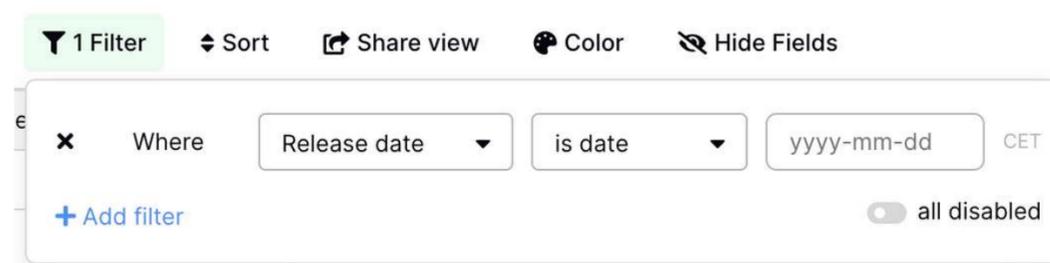
The screenshot shows a dialog box titled "You have not yet created a filter". Below the title, it says "Filters allow you to show rows that apply to your conditions." At the bottom left of the dialog, there is a blue "+ Add filter" button.




The screenshot shows the filter configuration interface. It has a header with "1 Filter", "Sort", "Share view", "Color", and "Hide Fields". Below the header, there is a filter rule: "Where Title is". There is an empty input field for the value. Below the rule, there is a "+ Add filter" button and a toggle switch labeled "all disabled".



Select the field where you apply the filter by changing **Title** into **Release date**.



The screenshot shows the filter configuration interface after the field has been changed. The filter rule now reads: "Where Release date is date". The input field contains "yyyy-mm-dd" and "CET". Below the rule, there is a "+ Add filter" button and a toggle switch labeled "all disabled".



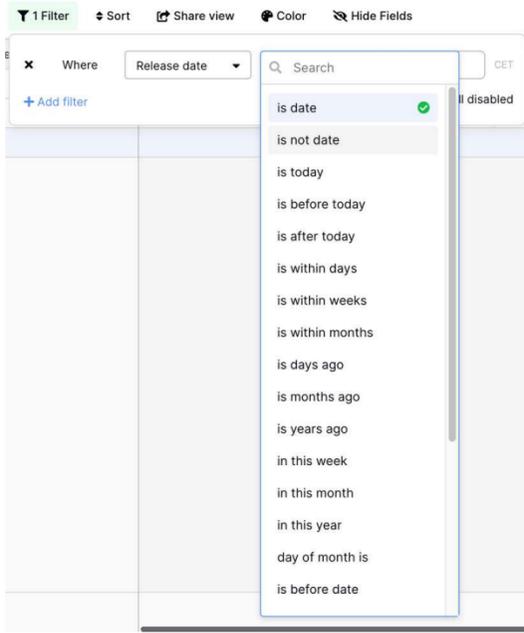
# 10.5 Add filters in a table

 Where

 Click on the arrow to open the **list of filtering rules available for Release Date.**



**Select the filtering rule**





- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
-



**Set the date to is after date.**  
 The last input field (where you set the date) stays empty.

 Where

 Add filter  all disabled





 **1 Filter** Your filter is created in the view.

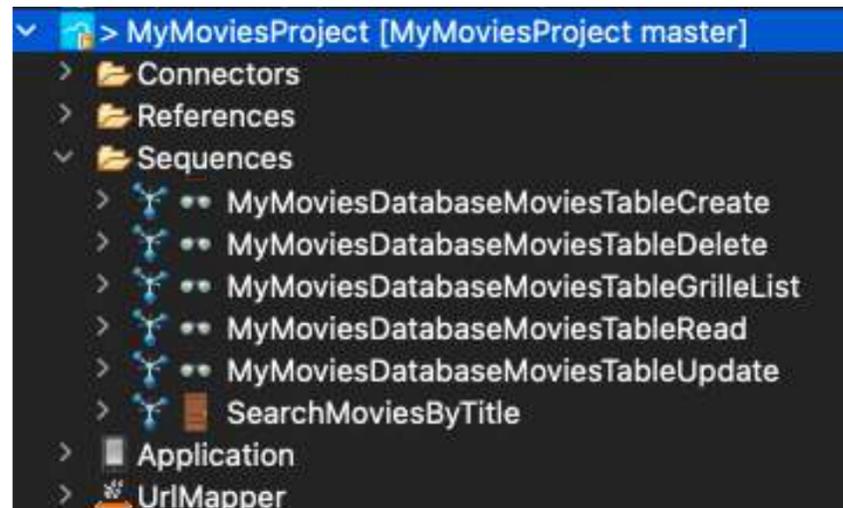
 WithFilter  **1 Filter**  Sort  Share view  Color  Hide Fields

	A Title	Overview	Release date	A Original title	A Image url
1					
2					
+					

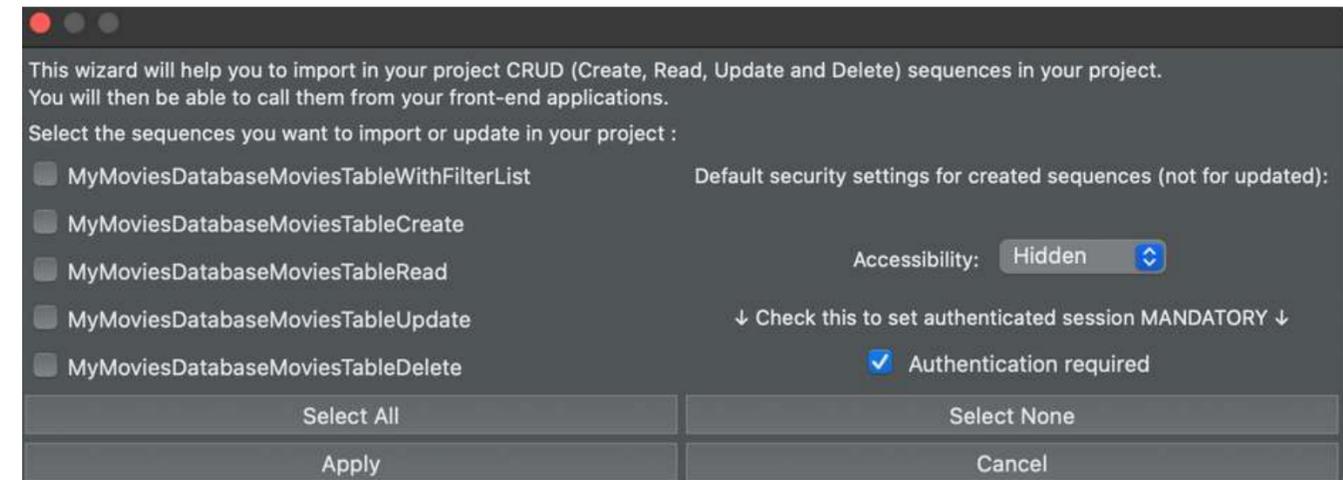


# 10.5 Add filters in a table

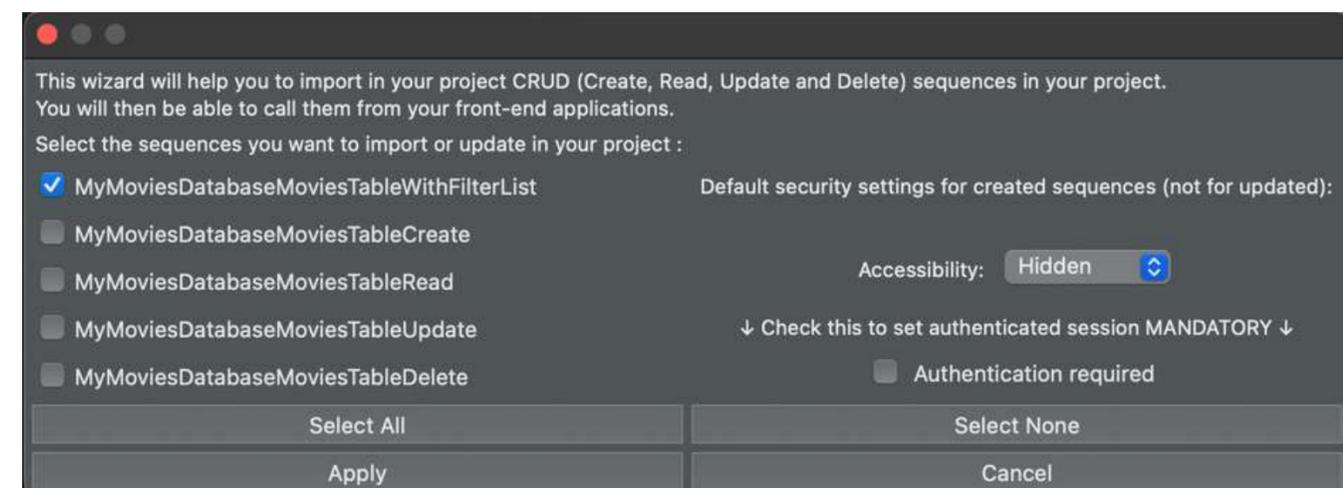
Let's create and import the sequence corresponding to the filter in **MyMoviesProject**.



In the **NoCode Databases** view, click on **Import into "MyMoviesProject"**.

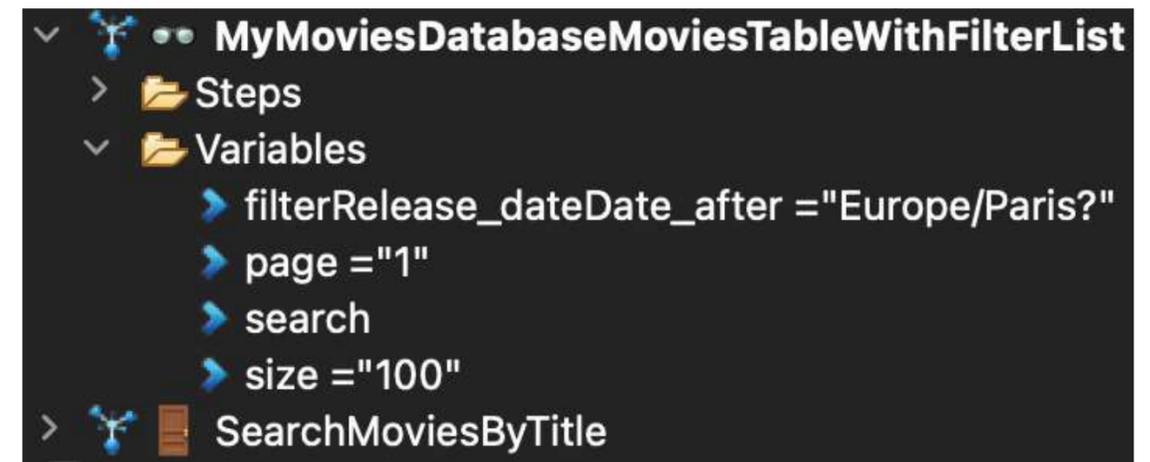
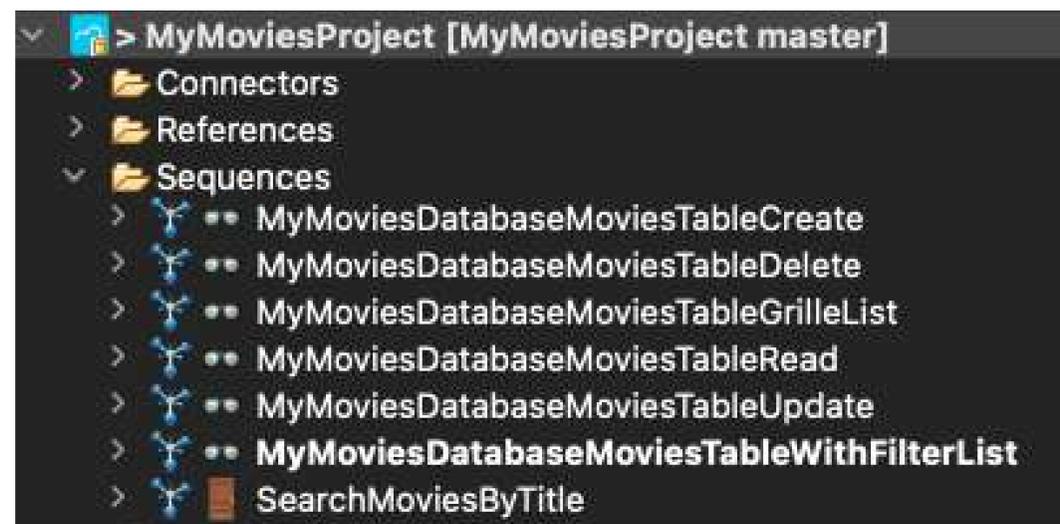


Select the sequence, uncheck **Authentication required** and click on **Apply**.



## 10.5 Add filters in a table

The sequence corresponding to the filter has been imported in **MyMoviesProject**.



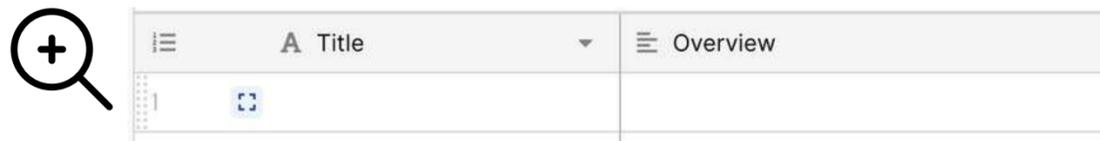
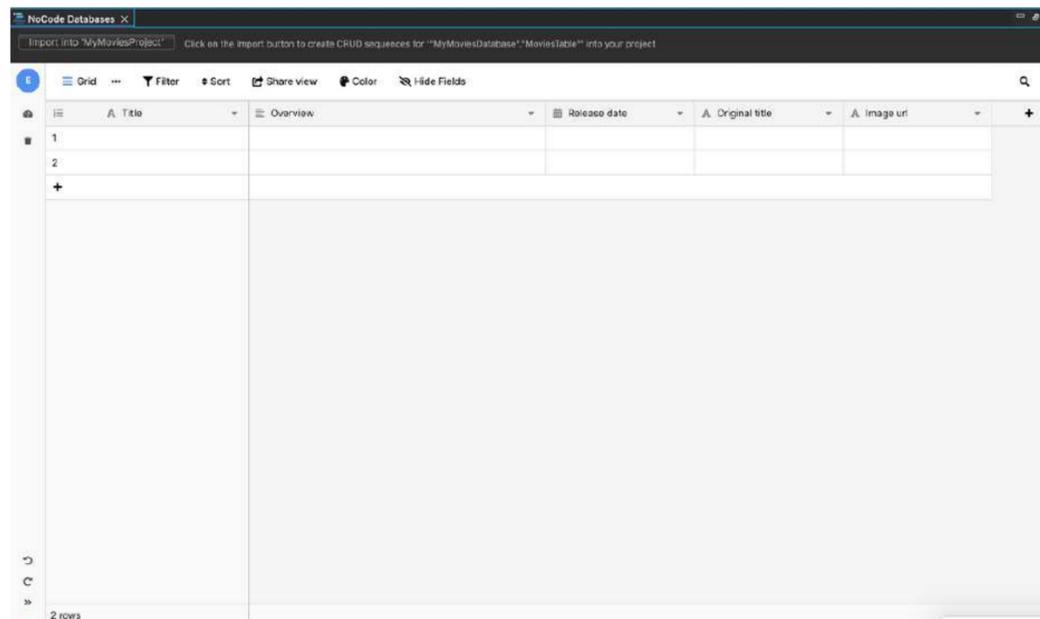
In the variables folder,  
we can see a variable corresponding to the filter.



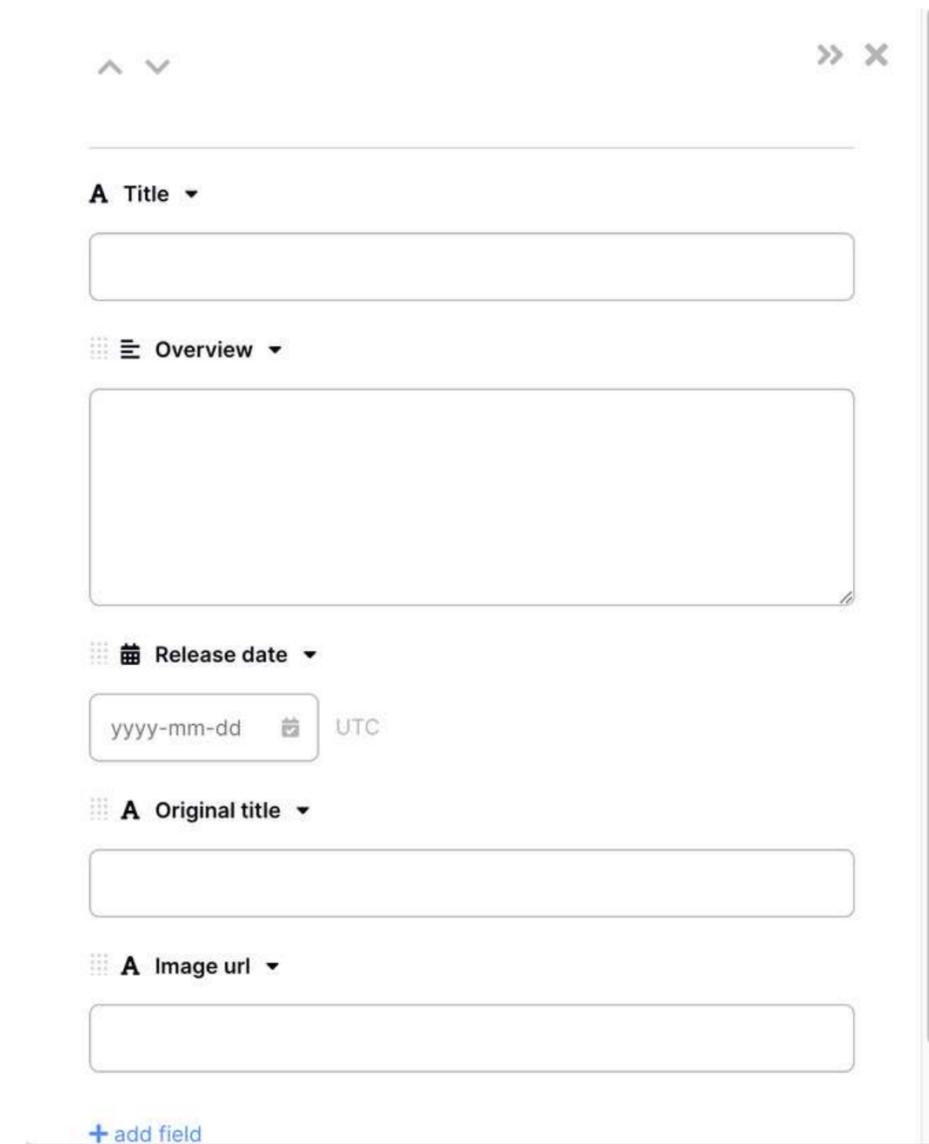
# 10.6 Test the CRUD sequences

Let's say we want to **read data from the first row** in MoviesTable.

First, let's create an entry directly in the table.



Click on this icon in the field Title to open the field editor

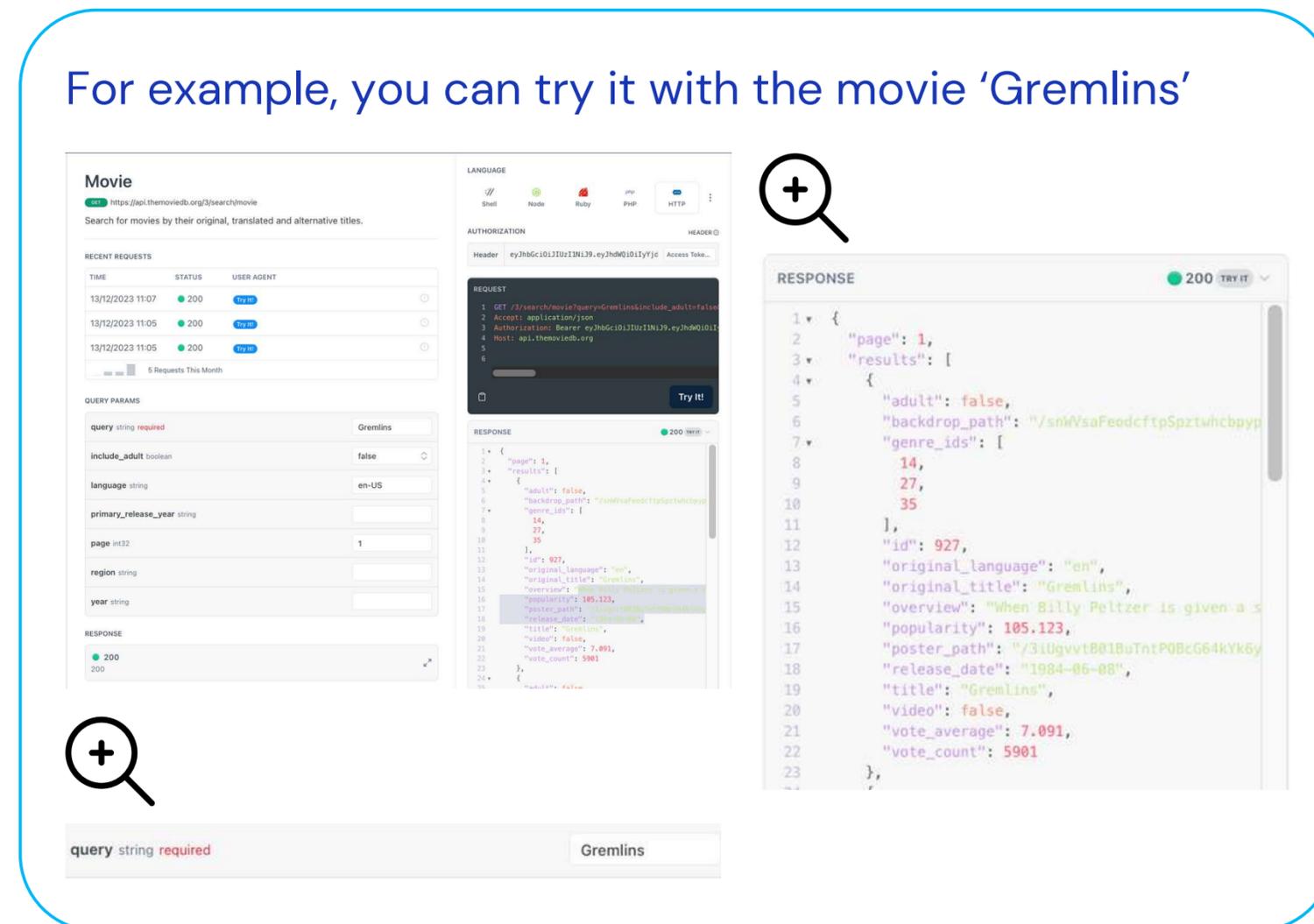


# 10.6 Test the CRUD sequences

You can search data in the editor available in the TMDb API Search Movie page.  
<https://developer.themoviedb.org/reference/search-movie>

Fill the fields in the editor.

For example, you can try it with the movie 'Gremlins'



The screenshot shows the TMDb API Search Movie editor interface. On the left, the 'QUERY PARAMS' section has 'query string required' set to 'Gremlins'. The 'RESPONSE' section shows a 200 status code. The main area displays the JSON response for the search query. A magnifying glass icon is positioned over the response area.

```

1 {
2   "page": 1,
3   "results": [
4     {
5       "adult": false,
6       "backdrop_path": "/snWsaFeodcftpSpztwhcbpyp",
7       "genre_ids": [
8         14,
9         27,
10        35
11      ],
12      "id": 927,
13      "original_language": "en",
14      "original_title": "Gremlins",
15      "overview": "When Billy Peltzer is given a strange but adorable pet named Gizmo for Christmas, he inadvertently breaks the three important rules of caring for a Mogwai, unleashing a horde of mischievous gremlins on a small town.",
16      "popularity": 105.123,
17      "poster_path": "/3iUgvvtB01BuTntPOBcG64kYk6y",
18      "release_date": "1984-06-08",
19      "title": "Gremlins",
20      "video": false,
21      "vote_average": 7.091,
22      "vote_count": 5901
23    }
24  ]
25 }
  
```



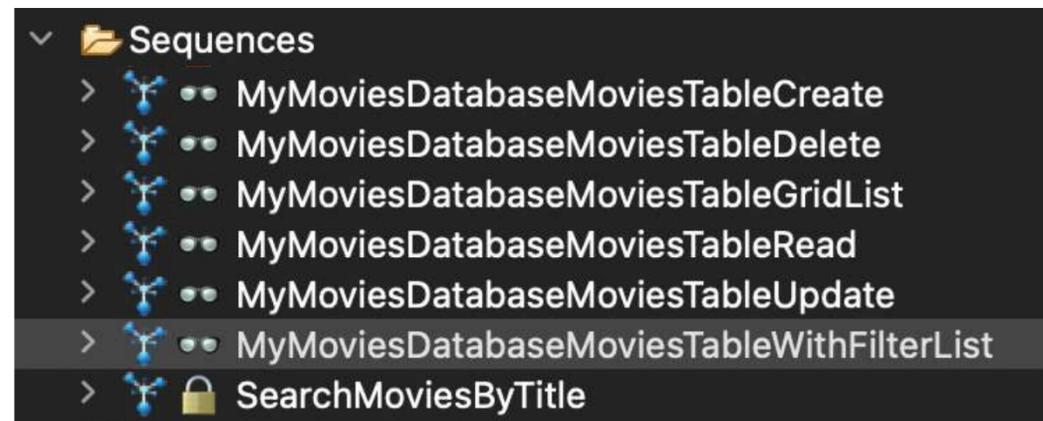
Now, we have data in the first row in MoviesTable

	A Title	Overview	Release date	A Original title	A Image url
1	Gremlins	When Billy Peltzer is given a strange but adorable pet name...	1984-06-08	Gremlins	https://image.tmbd.org/t/p/...
2					

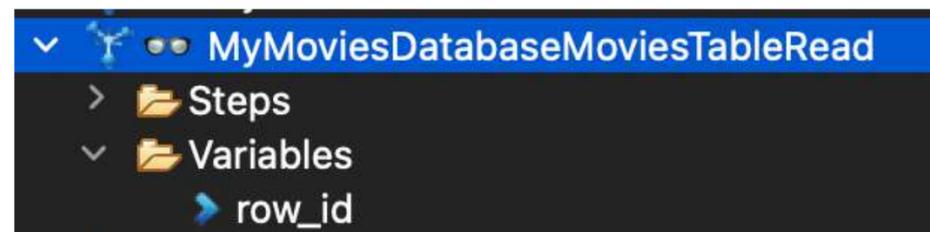


# 10.6 Test the CRUD sequences

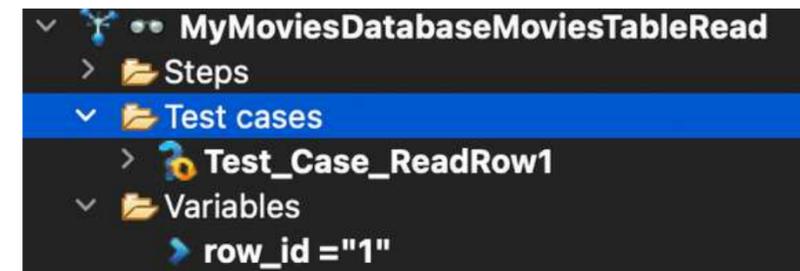
To read data from a row in MoviesTable, we use the sequence **MyMoviesDatabaseMoviesTableRead**.



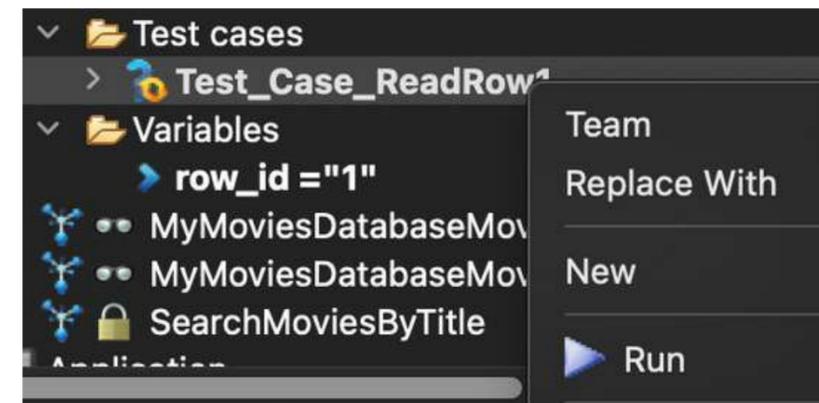
The sequence has one variable : row\_id



Let's create a test case with 1 as row\_id value



Run the test case



# 10.6 Test the CRUD sequences

The sequence returns an error: invalid token.

```
NoCode Databases | MyMoviesProject [S: MyMoviesDatabaseMoviesTableRead].json X
1 |
2 | "object": {
3 |   "detail": "Invalid token header. No token provided.",
4 |   "error": "ERROR_INVALID_TOKEN_HEADER"
5 | }
6 | }
```

If we look at the **Baserow\_API** response, we can see that the database returns an **error 401 Unauthorized**.

```
NoCode Databases | MyMoviesProject [S: MyMoviesDatabaseMoviesTableRead].json | lib_Baserow [C: Baserow_API_spec].json X
HTTP result (ContentType: application/json, Length: 90)
{"detail":"Invalid token header. No token provided.",error:"ERROR_INVALID_TOKEN_HEADER"}
1 | {
2 |   "HttpInfo": {
3 |     "url": "https://baserow-backend.convertigo.net/api/database/rows/table/1323/1/?us",
4 |     "status": {
5 |       "text": "",
6 |       "attr": {
7 |         "code": "401",
8 |         "text": "Unauthorized"
9 |       }
10 |    },
11 |     "headers": {
12 |       "header": {
13 |         "text": "",
14 |         "attr": {
15 |           "name": "Accept",
16 |           "value": "application/json"
17 |         }
18 |       },
19 |       "text": "",
20 |       "attr": {
21 |         "name": "Content-Type",
22 |         "value": "application/x-www-form-urlencoded"
23 |       }
24 |     },
25 |     "text": "",
26 |     "attr": {
27 |       "name": "Authorization",
28 |       "value": "Token "
29 |     }
30 |   },
31 |   "text": "",
32 |   "attr": {
33 |     "name": "User-Agent",
34 |     "value": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
35 |   }
36 | },
37 | "responseheaders": {
38 |   "header": {
39 |     "text": "",
40 |     "attr": {
41 |       "name": "Date",
42 |       "value": "Wed, 13 Dec 2023 18:48:46 GMT"
43 |     }
44 |   }
45 | }
46 | }
47 | }
48 | }
49 | }
50 | }
```



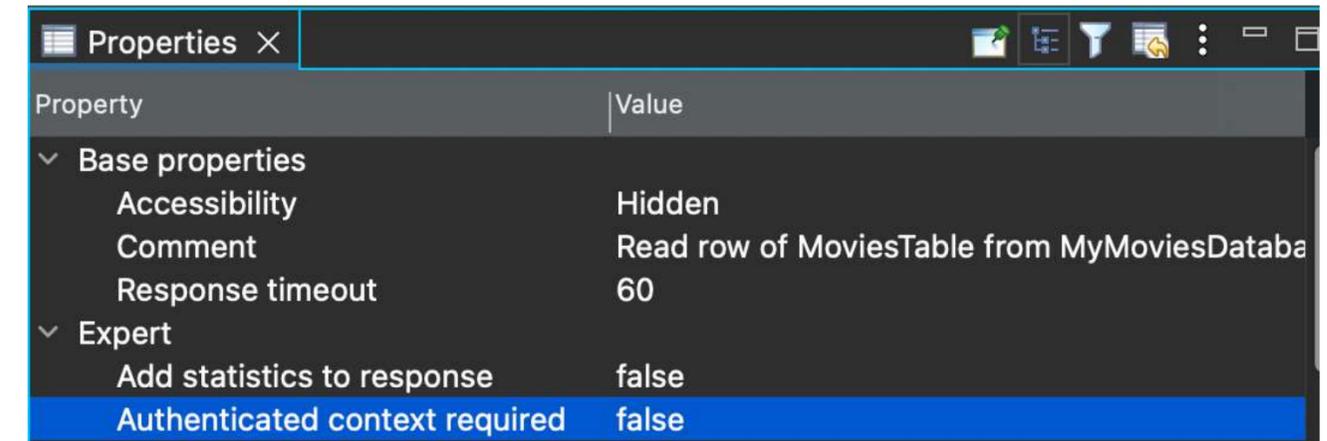
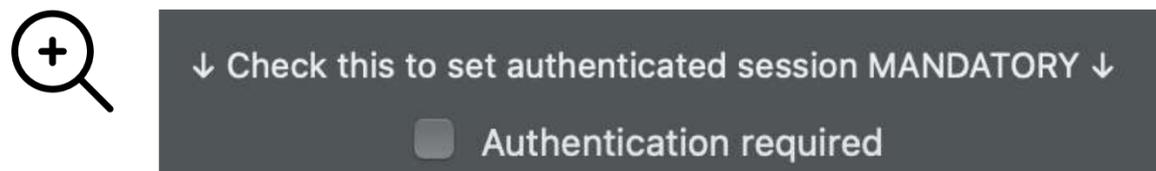
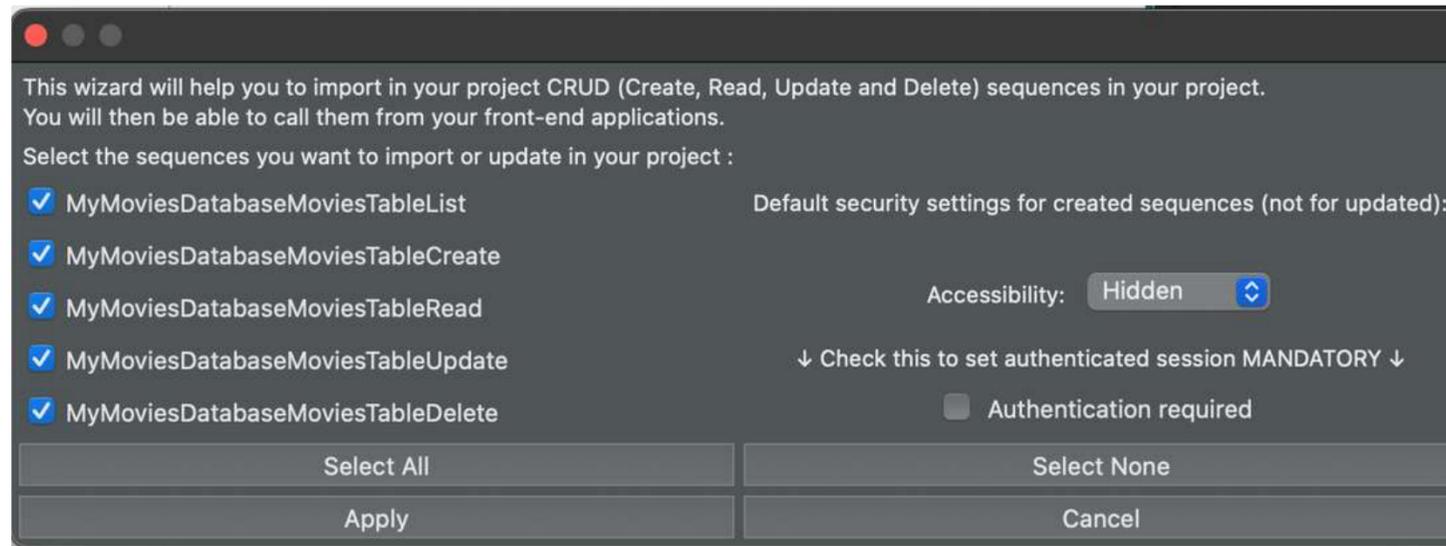
```
C: Baserow_API_spec].json X
1 | {
2 |   "HttpInfo": {
3 |     "url": "https://baserow-backend.convertigo.net/api/database/rows/table/1323/1/?us",
4 |     "status": {
5 |       "text": "",
6 |       "attr": {
7 |         "code": "401",
8 |         "text": "Unauthorized"
9 |       }
10 |    }
11 |  }
```



# 10.6 Test the CRUD sequences

Remember,  
when we created and imported the CRUD sequences,  
we unchecked the field **uncheck Authentication required**.

In the sequence properties,  
the **Authentication context required property**  
value is **false**, as expected.

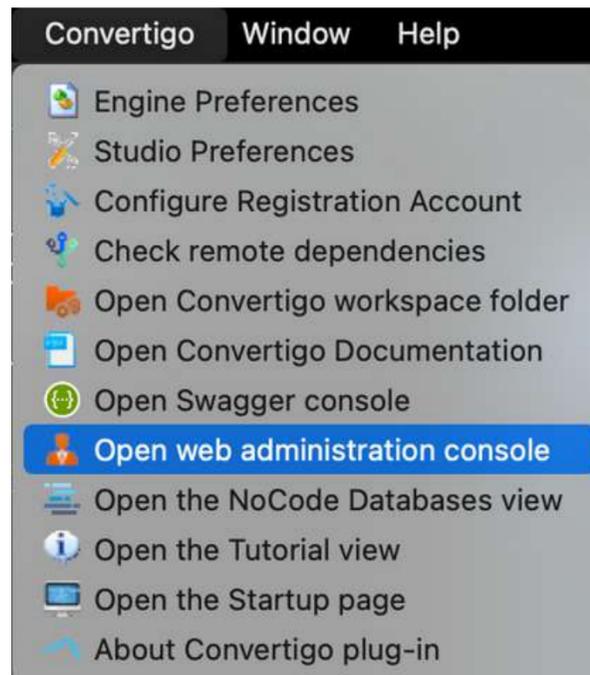


If we don't need an authentication, why does the Baserow\_API ask for a token ?

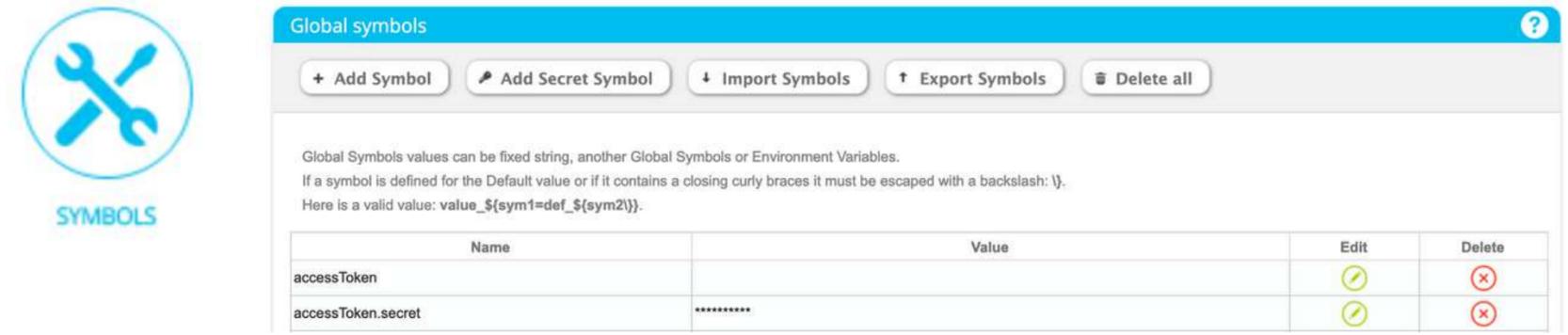


# 10.6 Test the CRUD sequences

The answer is in the Web administration console.



Let's look at the **Global symbols**.



**SYMBOLS**

Global symbols management interface showing a table of symbols:

Name	Value	Edit	Delete
accessToken			
accessToken.secret	*****		

Global Symbols values can be fixed string, another Global Symbols or Environment Variables.  
 If a symbol is defined for the Default value or if it contains a closing curly braces it must be escaped with a backslash: \}.  
 Here is a valid value: value\_\${sym1=def\_\${sym2}}.

**Baserow adds automatically an apikey in the symbols.**

lib_baserow.apikey.secret	*****		
---------------------------	-------	--	--

lib\_baserow.apikey.secret



# 10.6 Test the CRUD sequences

We need to delete this apikey.

lib_baserow.apikey.secret	*****		
---------------------------	-------	---	---



Click on the Delete icon.



**Confirmation** ✕

Do you really want to delete the symbol 'lib\_baserow.apikey.secret'?



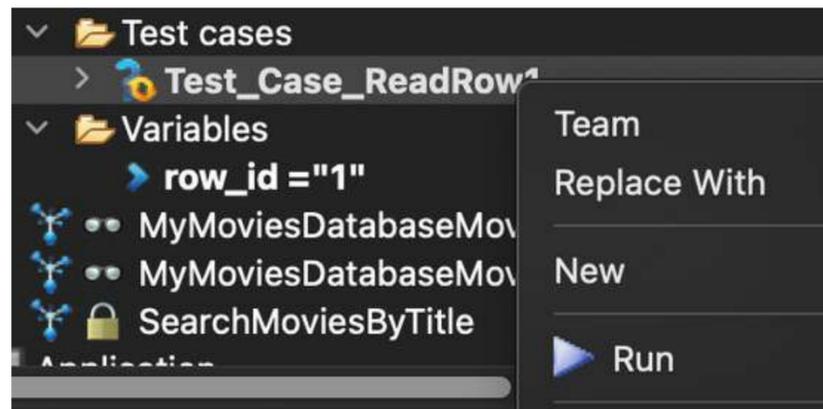
**Information** ✕

Global symbol 'lib\_baserow.apikey.secret' have been successfully deleted!



# 10.6 Test the CRUD sequences

Close the No Code database view and run the test case again.



The request is authorized by Baserow

lib\_BaseRow [C: Baserow\_API\_spec].json

```
{
  "HttpInfo": {
    "url": "https://baserow-backend.convertigo.net/api/database/rows/ta",
    "status": {
      "text": "",
      "attr": {
        "code": "200",
        "text": "OK"
      }
    }
  }
}
```



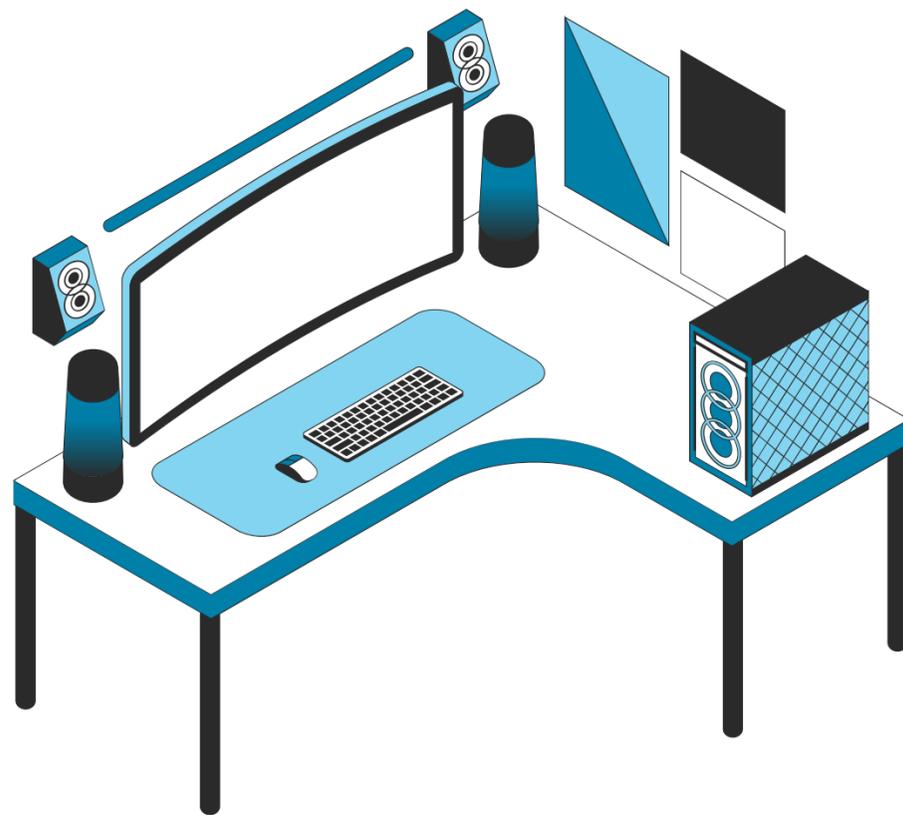
The sequence returns the data from the table.

```
1 | {
2 |   "object": {
3 |     "id": 1,
4 |     "order": "1.000000000000000000",
5 |     "Title": "Gremlins",
6 |     "Overview": "When Billy Peltzer is given a strange but adorable pet named Gizmo for",
7 |     "Release date": "1984-06-08",
8 |     "Original title": "Gremlins",
9 |     "Image url": "https://image.tmbd.org/t/p/w500/3iUgvvtB01BuTntP0BcG64kYk6y.jpg"
10 |   }
11 | }
```



# 11 – Logs

How to manage logs in the studio.



**11.1** Engine Log view

---

**11.2** Basics in Log level Configuration

---

**11.3** Configure Log level

---

**11.4** Logs in the web administration console

---

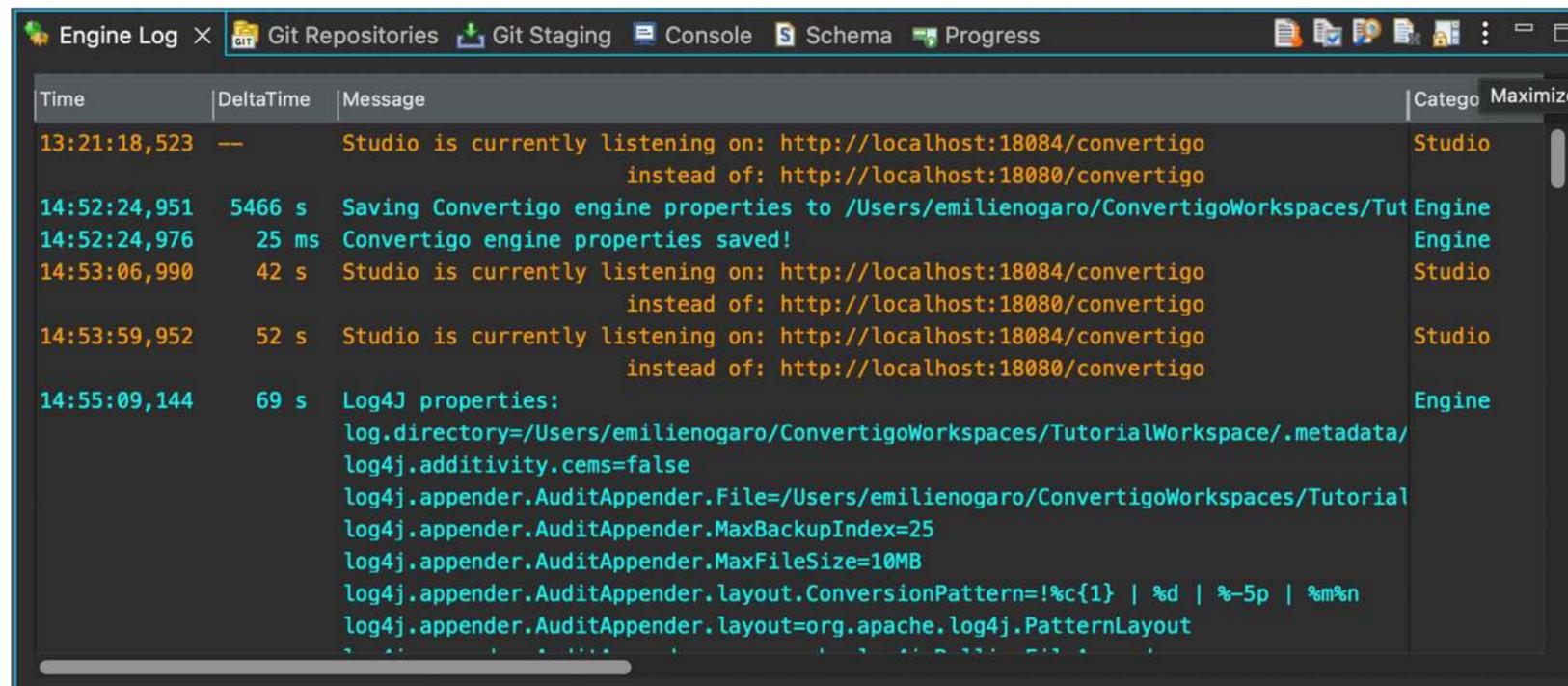
**11.5** Log step

---

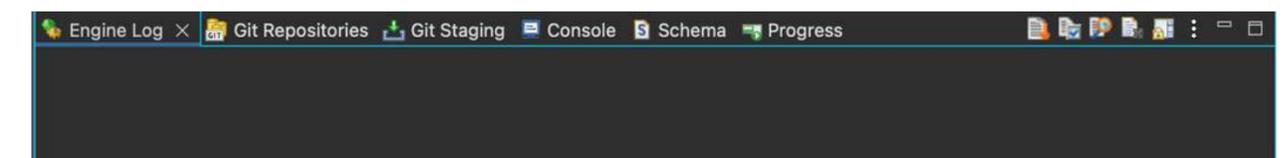
**11.6** Using Log step in a sequence

# 11.1 Engine Log view

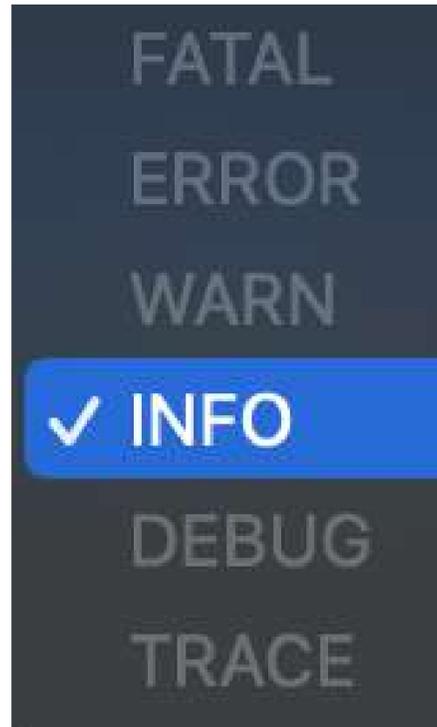
In the studio,  
the logs are displayed in the **Engine Log view**  
in the **Logs & Git panel**.



Click on the **Clear log viewer** icon  
to clear the logs.



## 11.2 Basics in Log level Configuration



**By default**, the **root logger** is set to **INFO**.

The others loggers are set to **WARN** or **INFO**.

The most useful and important logs (beans, context, engine and user context) are set to **“Inherited from root logger”**.

In **development mode**, to analyze issues in a project, the root logger is set to **DEBUG**.

In **Production mode**, we usually set **root logger** to **WARN**.

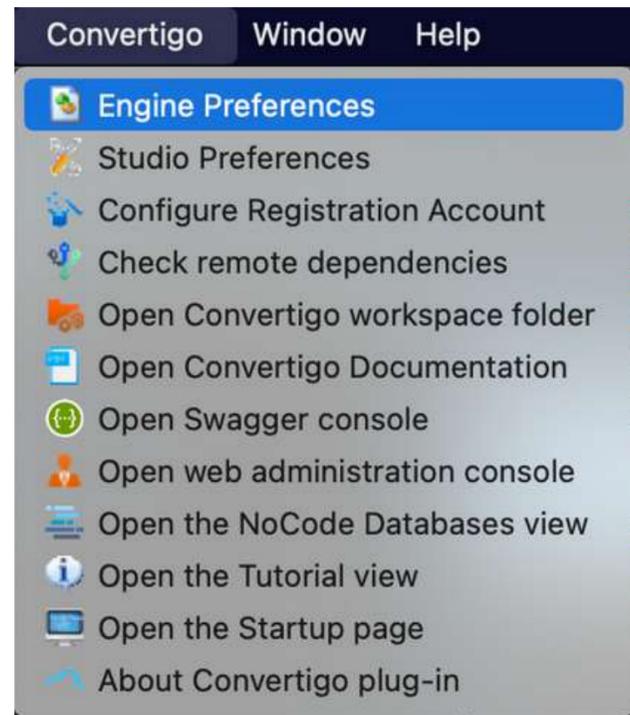
Sometimes, other loggers (specific and not commonly used) are lowered to **WARN** to gain space and speed.



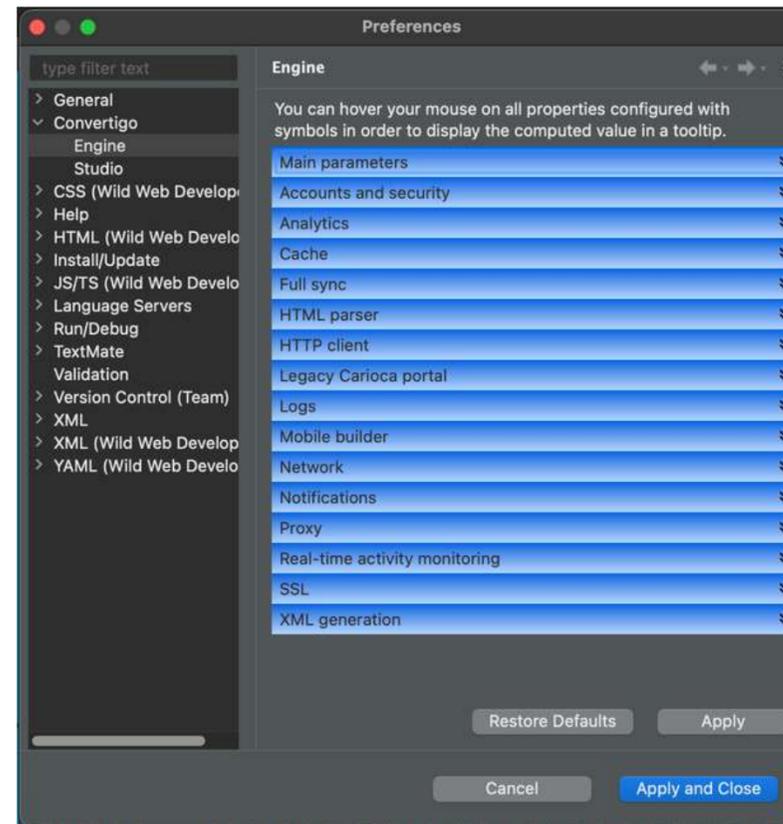
# 11.3 Configure Log level

In the studio, you can configure the log level in different places.

First option:  
Click on **Convertigo**,  
then select **Engine Preferences**



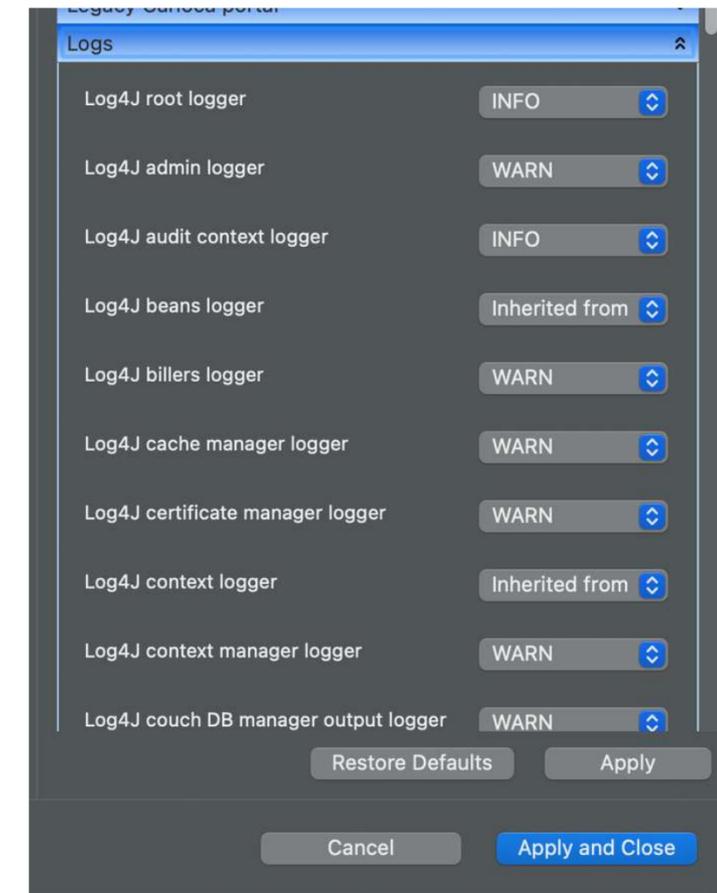
The **Preferences windows**  
appears.



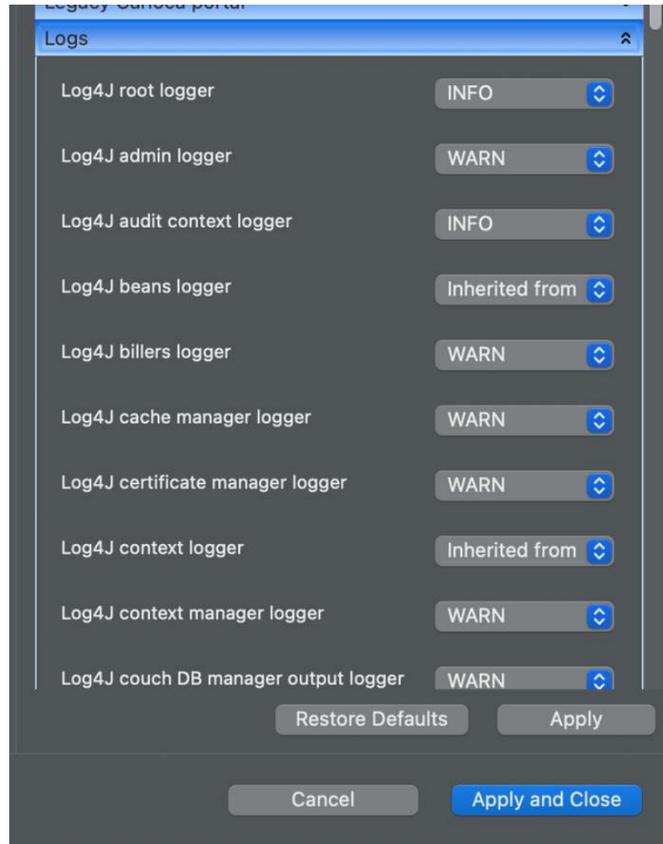
Click on the **Logs** tab,  
to open the logs settings



Logs



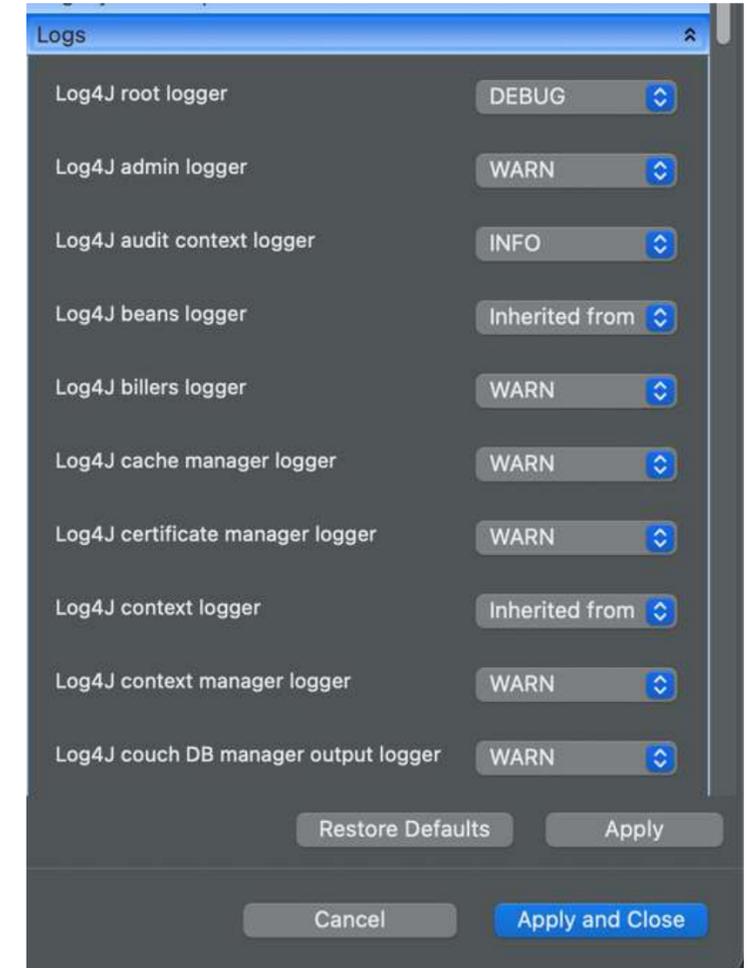
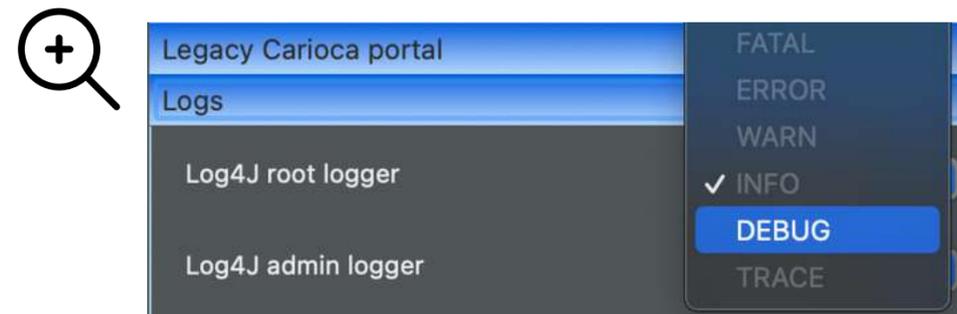
# 11.3 Configure Log level



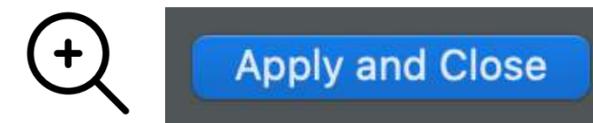
By default, the root logger is set on INFO.



Click on the select button to display the different log settings and select **DEBUG**.



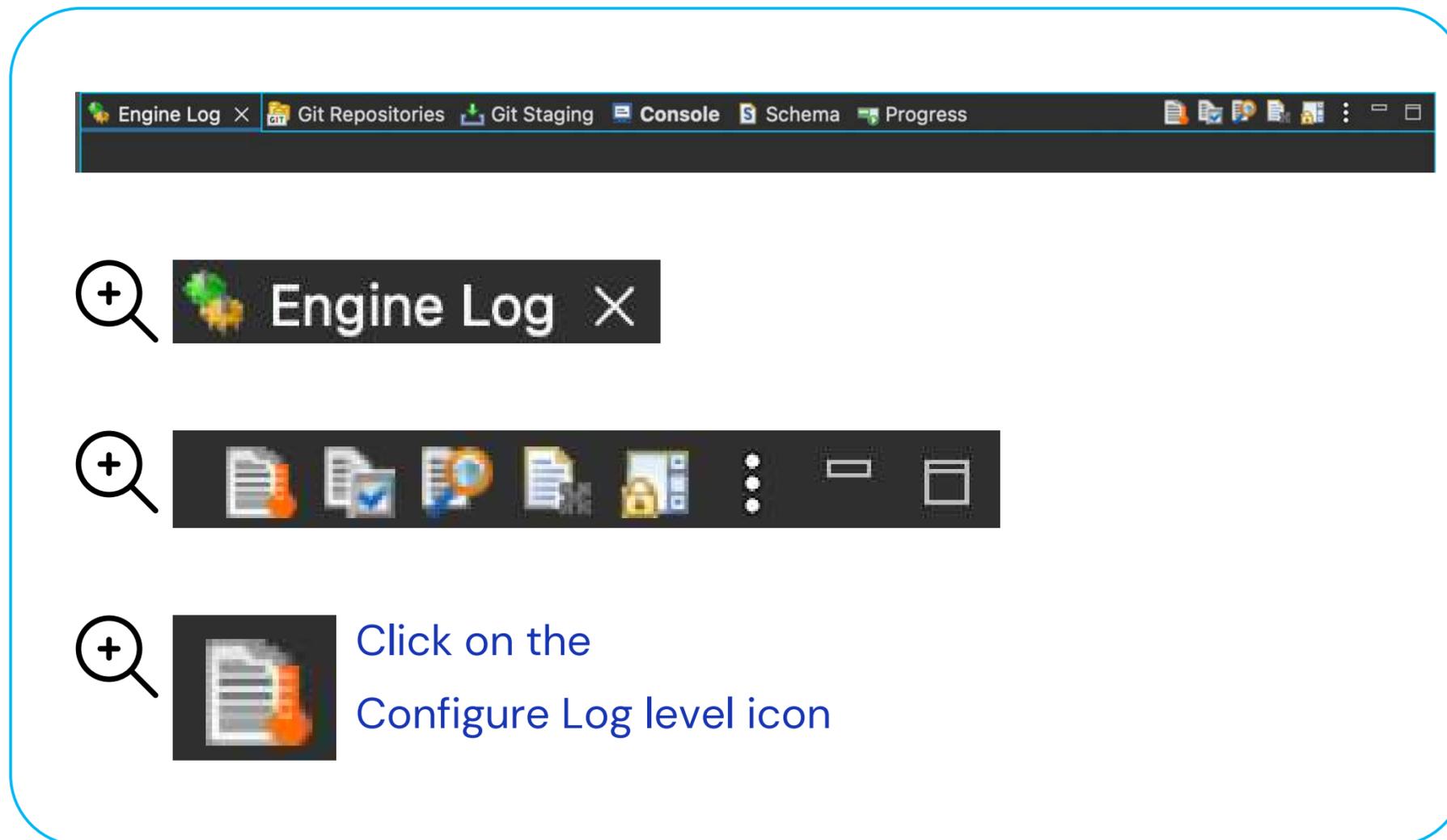
Click on Apply and Close.



# 11.3 Configure Log level

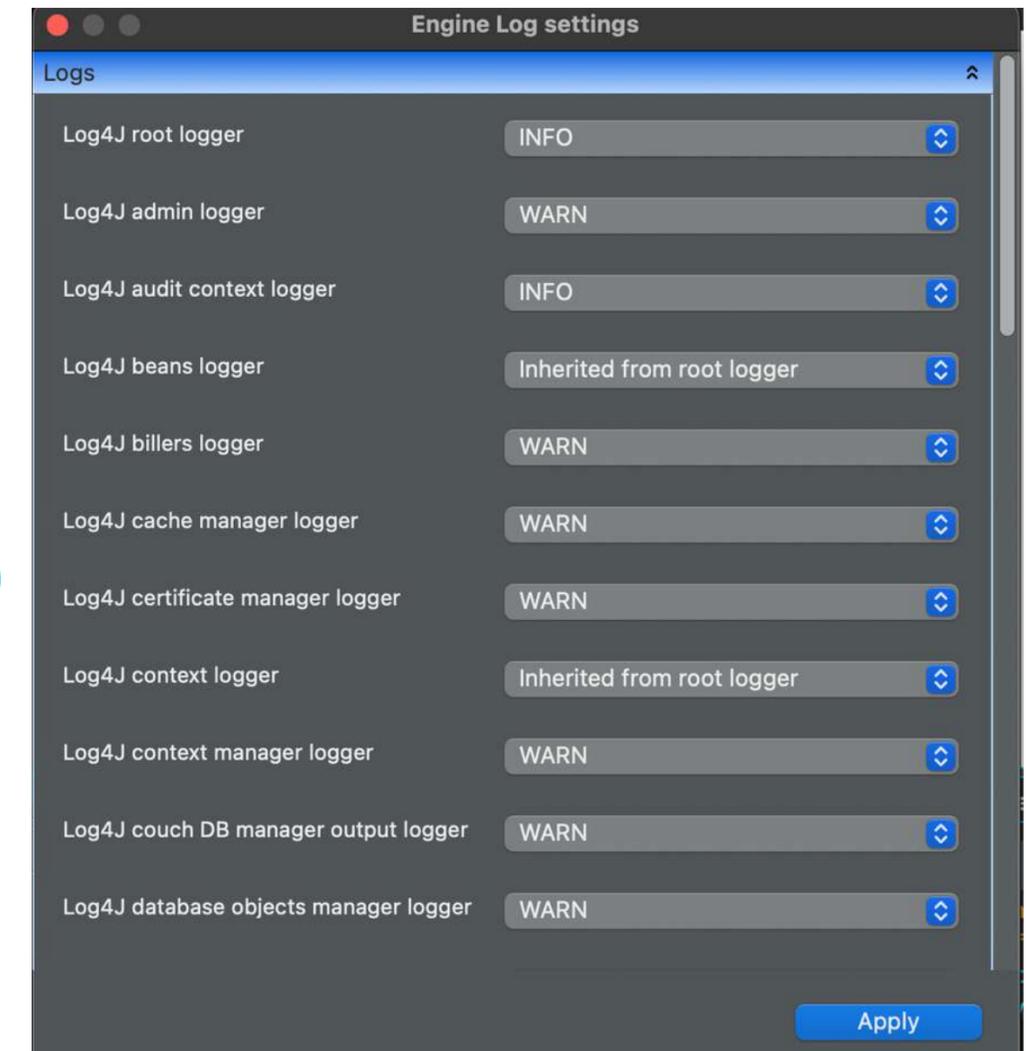
Second option:

In the **Engine Log view** in the **Logs & Git panel**



Click on the  
Configure Log level icon

The **Engine Log settings** window appears.



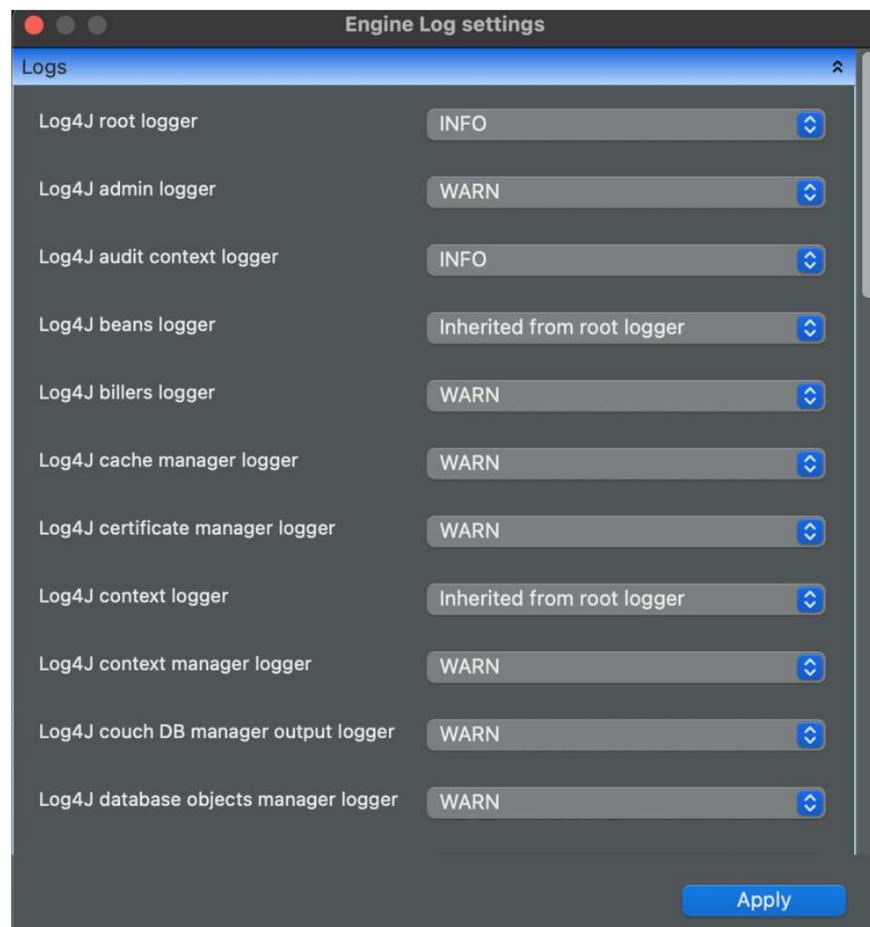
Logger	Log Level
Log4J root logger	INFO
Log4J admin logger	WARN
Log4J audit context logger	INFO
Log4J beans logger	Inherited from root logger
Log4J billers logger	WARN
Log4J cache manager logger	WARN
Log4J certificate manager logger	WARN
Log4J context logger	Inherited from root logger
Log4J context manager logger	WARN
Log4J couch DB manager output logger	WARN
Log4J database objects manager logger	WARN

Apply



# 11.3 Configure Log level

In the Engine Log settings window,



Let's look at the root logger.



Log4J root logger

INFO



Click on the select button to display the different log settings.



- FATAL
- ERROR
- WARN
- ✓ INFO
- DEBUG
- TRACE

→

Select **DEBUG**.

- FATAL
- ERROR
- WARN
- ✓ INFO
- DEBUG**
- TRACE

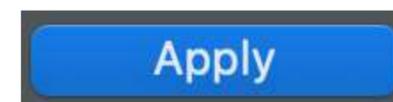


Log4J root logger

DEBUG



Click on Apply.



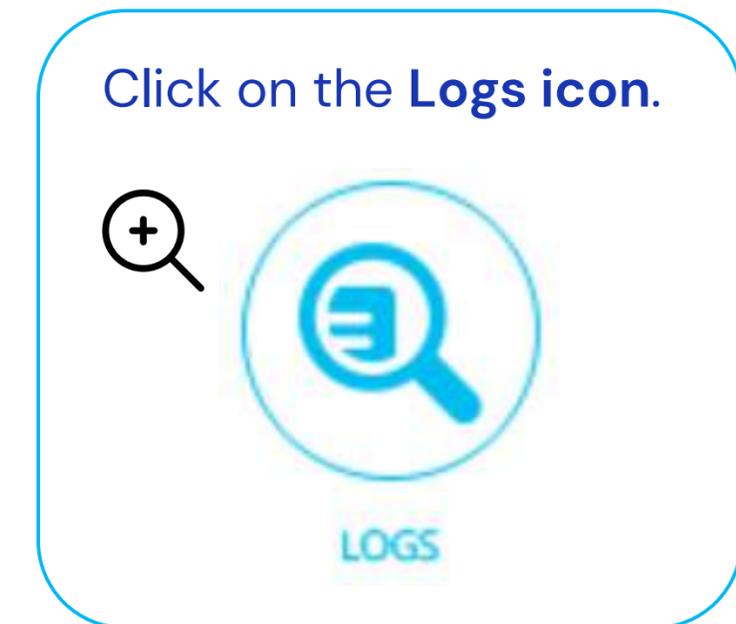
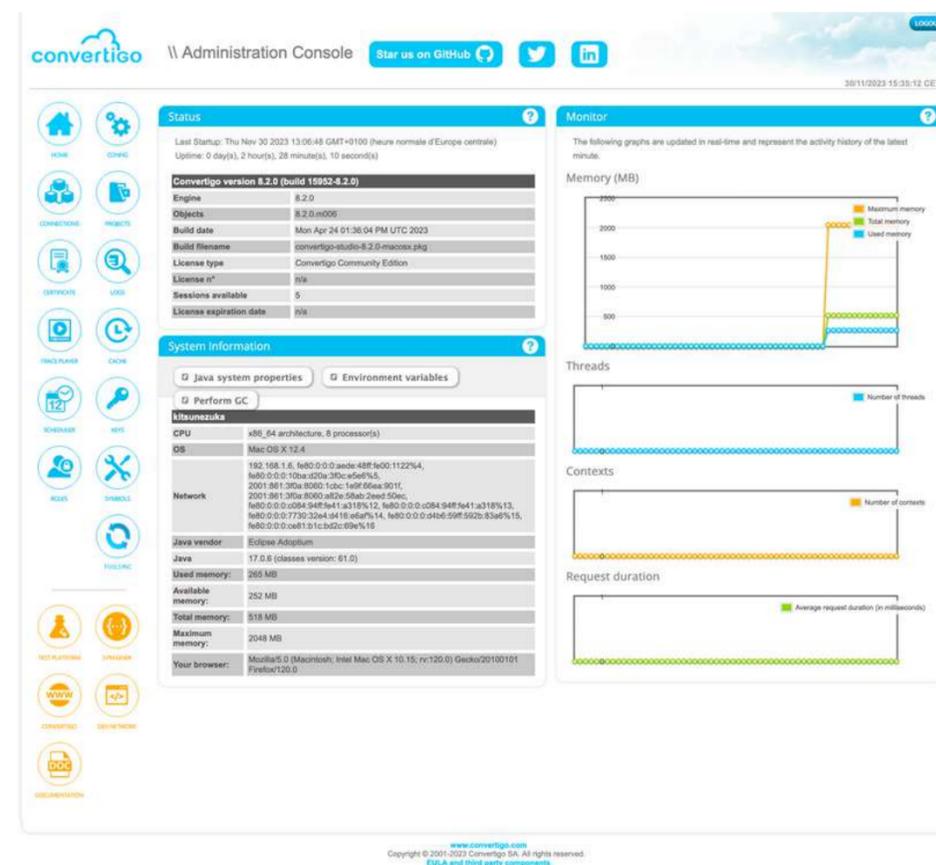
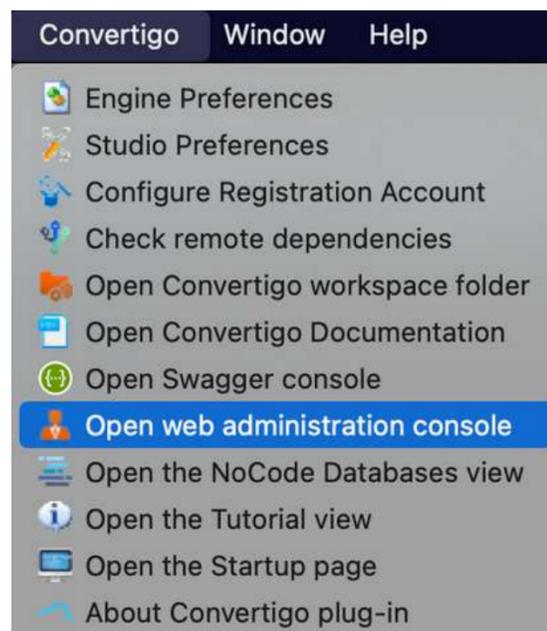
# 11.4 Logs in the web administration console



You can also configure and view the logs in the web administration console.

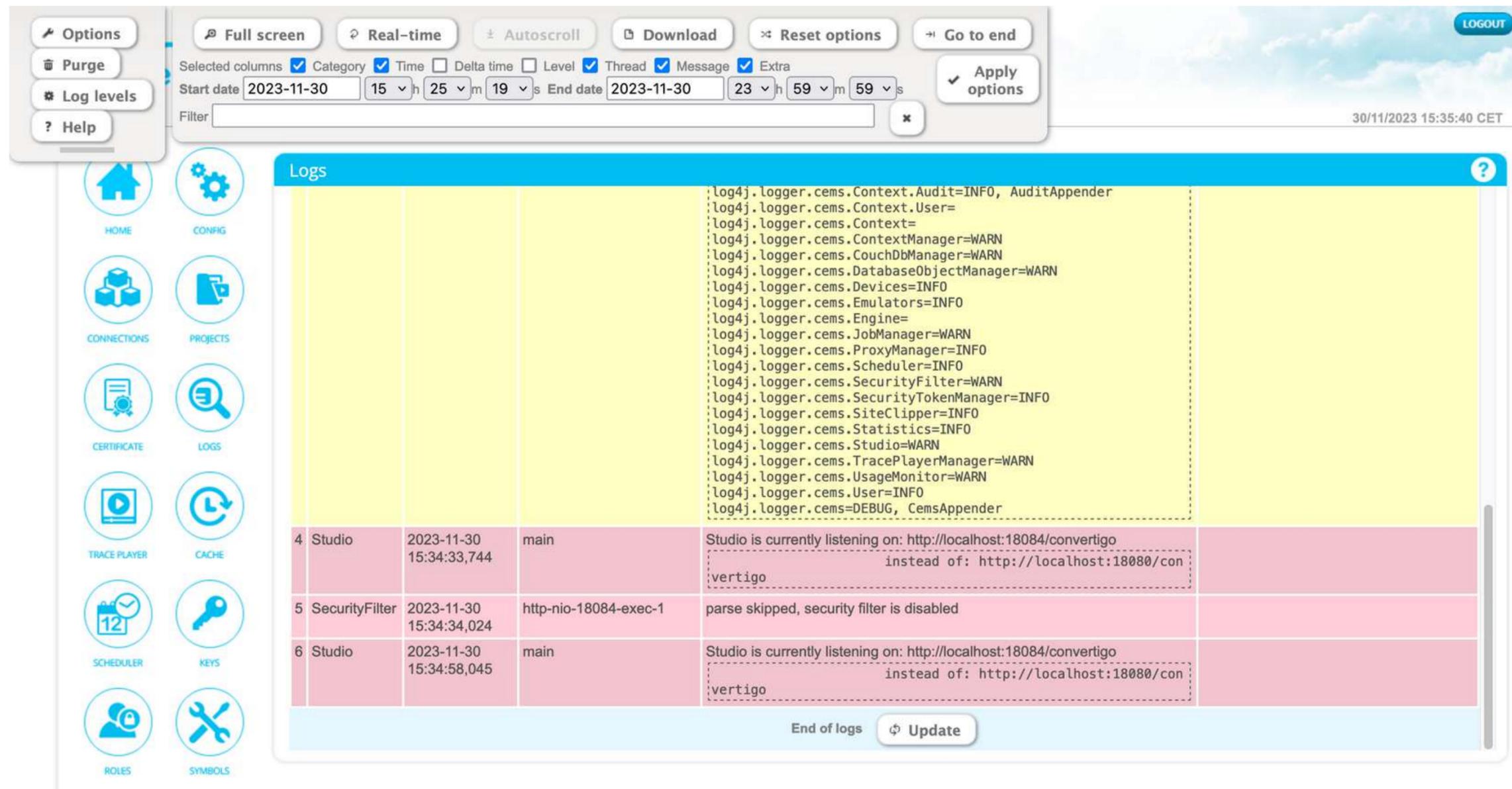
Open the **web administration console**.

In the web administration console



# 11.4 Logs in the web administration console

In the Logs page of the web administration console, you can see and filter the logs.



Options  
Purge  
Log levels  
Help

Full screen Real-time Autoscroll Download Reset options Go to end

Selected columns  Category  Time  Delta time  Level  Thread  Message  Extra

Start date 2023-11-30 15 h 25 m 19 s End date 2023-11-30 23 h 59 m 59 s

Filter

30/11/2023 15:35:40 CET

LOGOUT

HOME CONFIG CONNECTIONS PROJECTS CERTIFICATE LOGS TRACE PLAYER CACHE SCHEDULER KEYS ROLES SYMBOLS

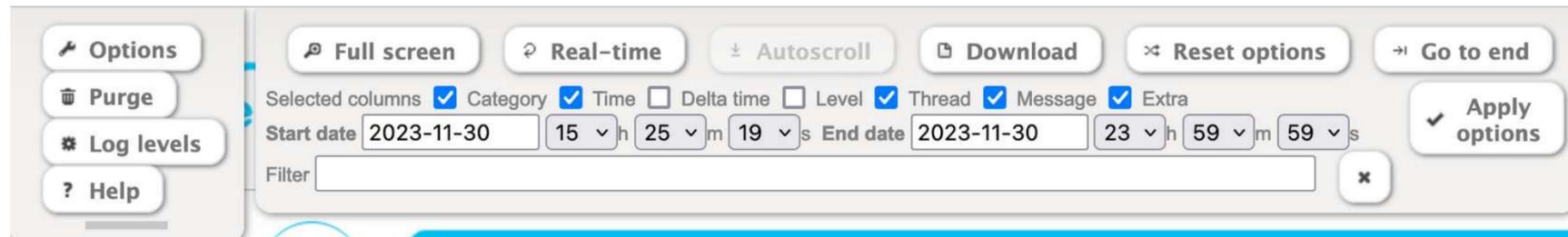
### Logs

				log4j.logger.cems.Context.Audit=INFO, AuditAppender log4j.logger.cems.Context.User= log4j.logger.cems.Context= log4j.logger.cems.ContextManager=WARN log4j.logger.cems.CouchDbManager=WARN log4j.logger.cems.DatabaseObjectManager=WARN log4j.logger.cems.Devices=INFO log4j.logger.cems.Emulators=INFO log4j.logger.cems.Engine= log4j.logger.cems.JobManager=WARN log4j.logger.cems.ProxyManager=INFO log4j.logger.cems.Scheduler=INFO log4j.logger.cems.SecurityFilter=WARN log4j.logger.cems.SecurityTokenManager=INFO log4j.logger.cems.SiteClipper=INFO log4j.logger.cems.Statistics=INFO log4j.logger.cems.Studio=WARN log4j.logger.cems.TracePlayerManager=WARN log4j.logger.cems.UsageMonitor=WARN log4j.logger.cems.User=INFO log4j.logger.cems=DEBUG, CemsAppender	
4	Studio	2023-11-30 15:34:33,744	main	Studio is currently listening on: http://localhost:18084/convertigo instead of: http://localhost:18080/con vertigo	
5	SecurityFilter	2023-11-30 15:34:34,024	http-nio-18084-exec-1	parse skipped, security filter is disabled	
6	Studio	2023-11-30 15:34:58,045	main	Studio is currently listening on: http://localhost:18084/convertigo instead of: http://localhost:18080/con vertigo	

End of logs Update

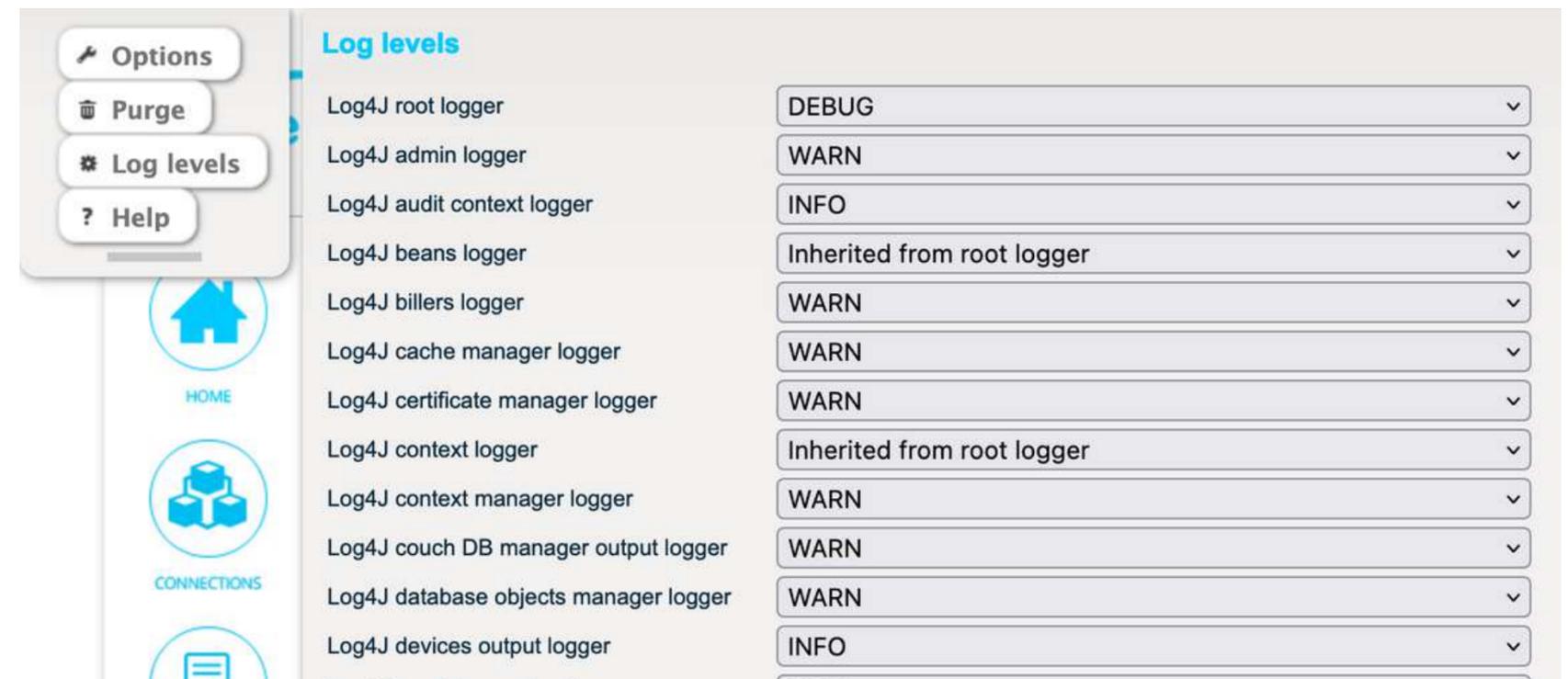


# 11.4 Logs in the web administration console

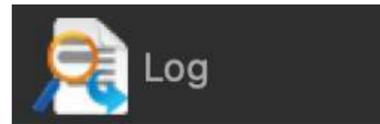


# Log levels

Click on **Log levels** to display a window where you can **configure the Log levels**.



# 11.5 Log step



## Log - Others step

This step **outputs data in a log file.**

More accurately, it outputs

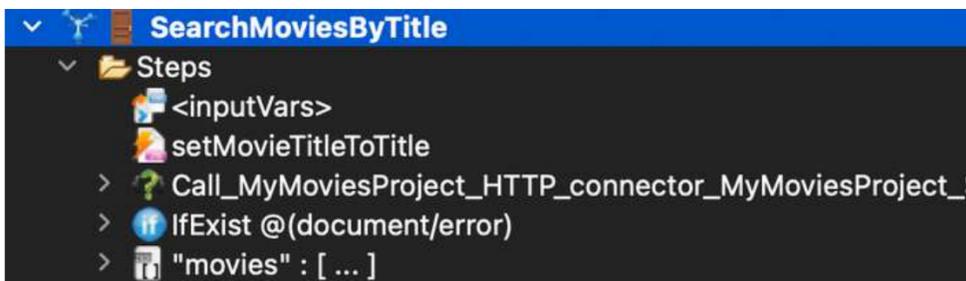
- a **message** (text string) generated from the **JavaScript expression** defined in **Expression property**
- in the **Convertigo logger** defined in the **Logger property**,
- for the **log level** defined in the **Level property**.

Property	Value
Base properties	
Comment	
Expression	//todo
Is active	true
Level	INFO
Logger	cems.Context



# 11.6 Using Log step in a sequence

Let's say we want to **log the inputVars** of the sequence **SearchMoviesByTitle** in the Engine Log.

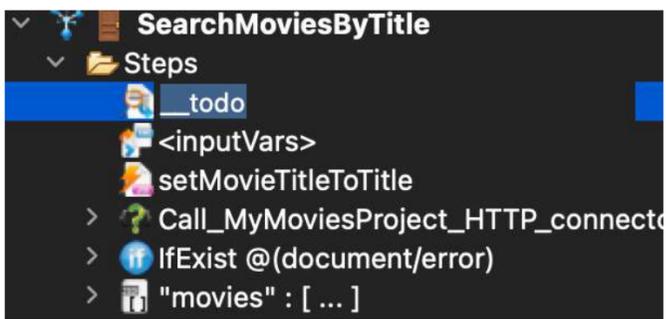


↓



Drag and drop a **Log step** in the sequence

↓



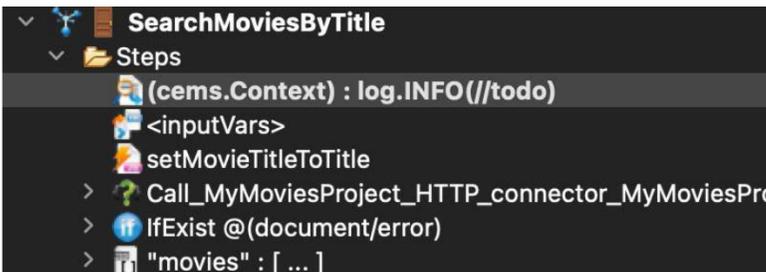



In the **Log step** properties,

- the **Level** is set to **INFO** (default)
- the **Logger** is set to **Context** (default)
- the **Expression** is yet to be defined

Property	Value
Base properties	
Comment	
Expression	//todo
Is active	true
Level	INFO
Logger	cems.Context

The log step appears like this in the treeview.

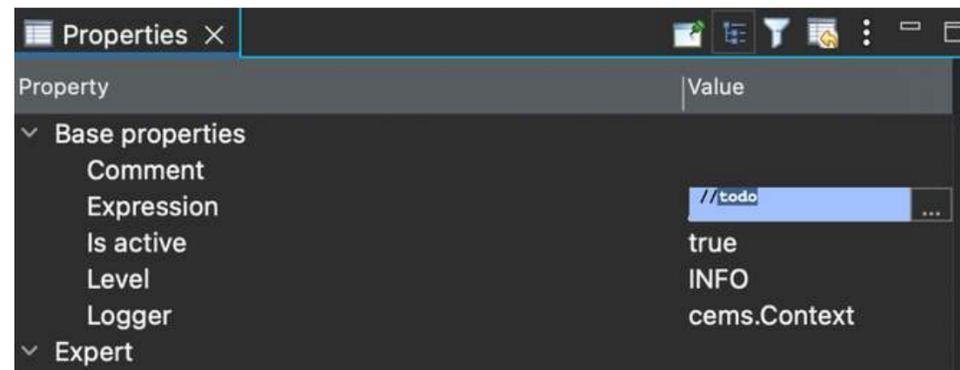
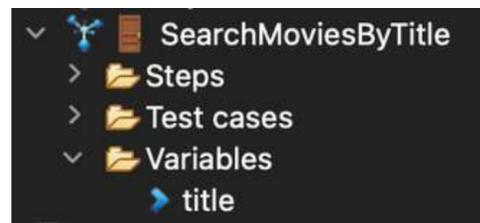




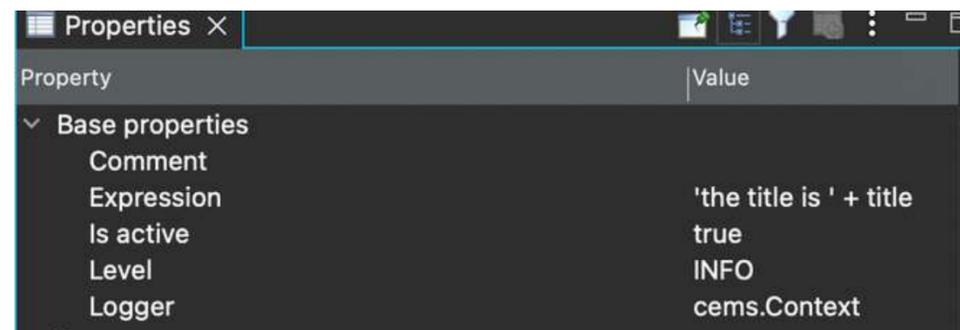
# 11.6 Using Log step in a sequence

Let's define the JavaScript expression.

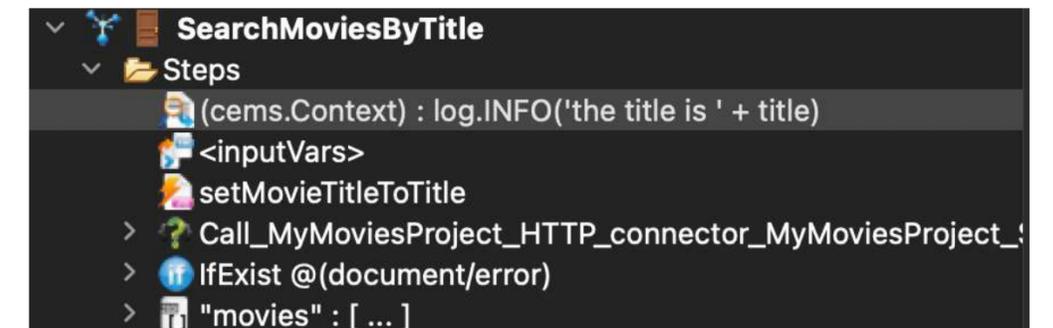
**title** is the **input variable** of the sequence.



In the **Expression** property, replace **//todo** by **'the title is ' + title**.

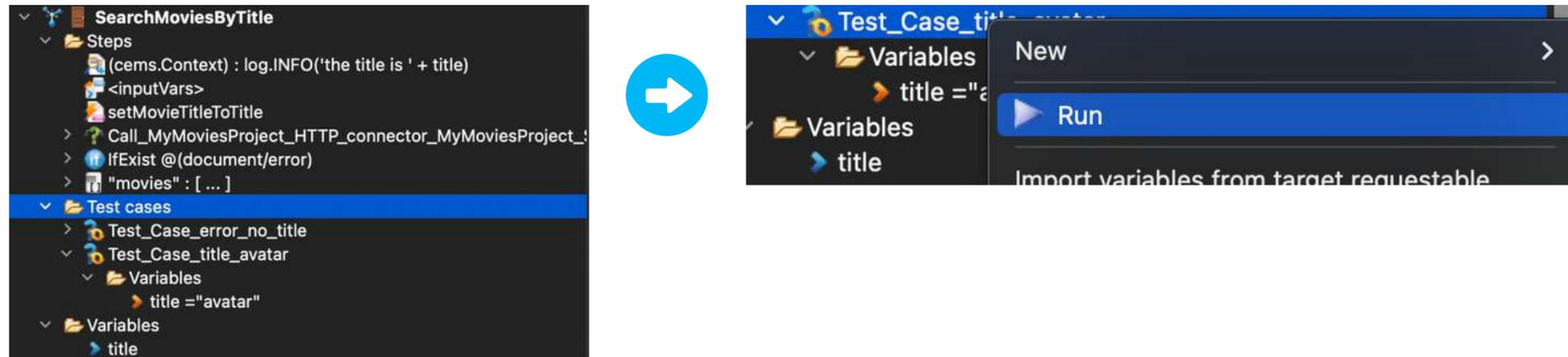


The log step appears like this in the treeview.

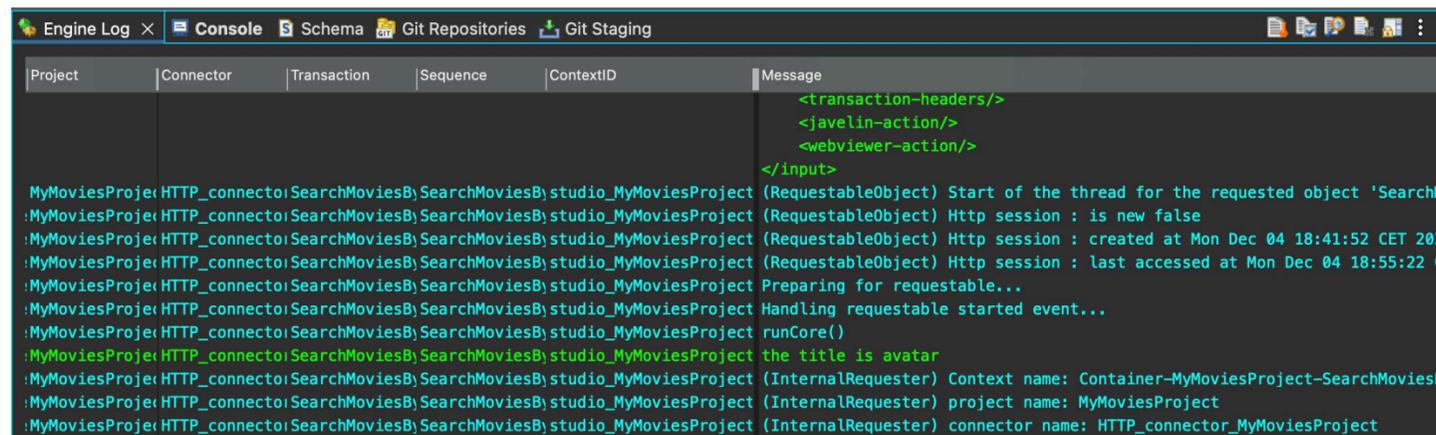


# 11.6 Using Log step in a sequence

Let's run the test case with avatar as title.



The image shows two screenshots of a test case configuration. The left screenshot shows a tree view for 'SearchMoviesByTitle' with a 'Test cases' folder containing 'Test\_Case\_title\_avatar'. Underneath, a 'Variables' folder contains 'title = "avatar"'. A blue arrow points to the right screenshot, which shows a 'New' dialog box with a 'Run' button and the text 'Import variables from target requestable'.

The screenshot shows the 'Engine Log' window with a table of log entries. The entry for 'the title is avatar' is highlighted in green. The table has columns for Project, Connector, Transaction, Sequence, ContextID, and Message.

Project	Connector	Transaction	Sequence	ContextID	Message
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	<transaction-headers/>
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	<javelin-action/>
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	<webviewer-action/>
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	</input>
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	(RequestableObject) Start of the thread for the requested object 'SearchMov
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	(RequestableObject) Http session : is new false
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	(RequestableObject) Http session : created at Mon Dec 04 18:41:52 CET 2023
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	(RequestableObject) Http session : last accessed at Mon Dec 04 18:55:22 CET
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	Preparing for requestable...
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	Handling requestable started event...
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	runCore()
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	the title is avatar
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	(InternalRequester) Context name: Container-MyMoviesProject-SearchMoviesBy
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	(InternalRequester) project name: MyMoviesProject
MyMoviesProje	HTTP_connecto	SearchMoviesB	SearchMoviesB	studio_MyMoviesProject	(InternalRequester) connector name: HTTP_connector_MyMoviesProject

In the Engine Log, the log the title is avatar appears in green (INFO level color).

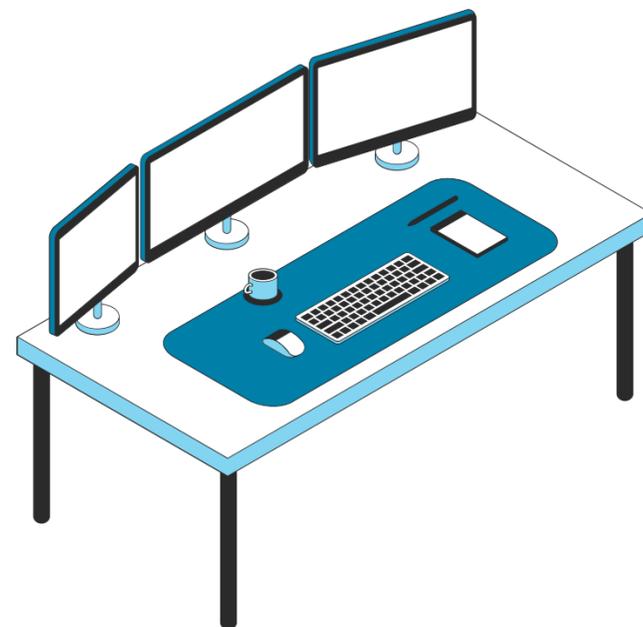


```
MyMoviesProje HTTP_connecto SearchMoviesB SearchMoviesB studio_MyMoviesProject the title is avatar
```



# 12 – Authentication

How to manage authentication  
in the studio.



**12.1** Accessibility property

---

**12.2** Http session management steps

---

**12.3** Create a Login sequence

---

**12.4** Test the Login sequence

---

**12.5** Add authentication to a sequence

---

# 12.1 Accessibility property

Sequences and transactions have a **property** called **Accessibility**.

It can take the following values:

- **Public:** The transaction/sequence is **runnable from everyone and everywhere**, and **visible in the Test Platform**.
- **Hidden:** It is **runnable** but **only** if you know the **execution URL**, and **not visible in the Test Platform**.
- **Private:** It is only **runnable from within the Convertigo engine** (Call Transaction/Call Sequence steps), and is **not visible in the Test Platform**.

This value is used for tests, unfinished transactions/sequences or functionalities not to be exposed.

**Private transactions/sequences remain runnable in the Studio**, for the developer to be able to test its developments.

Note: In the Test Platform:

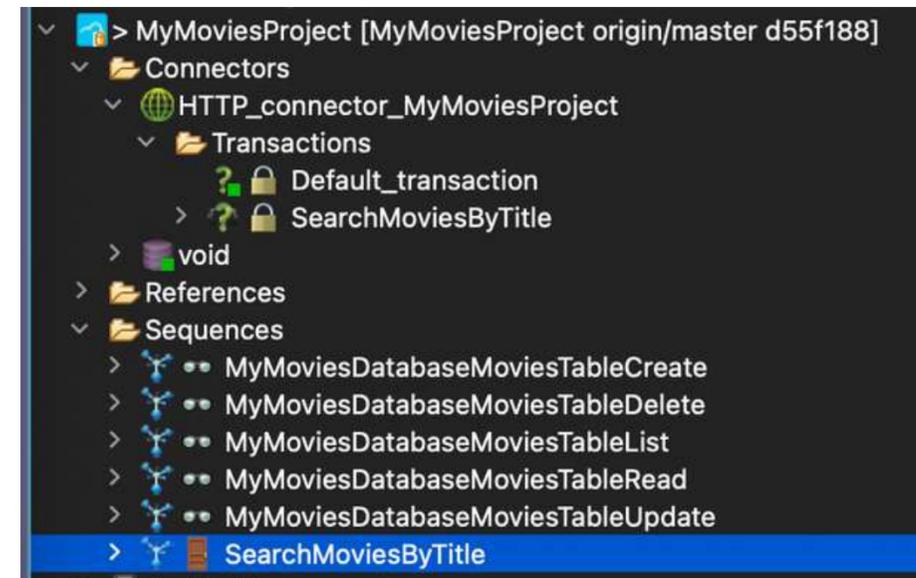
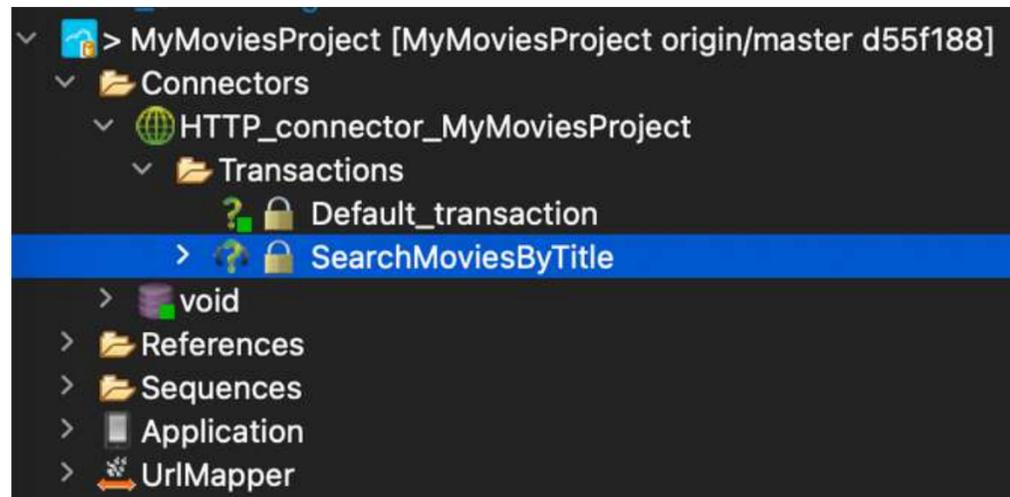
- The **administrator user** (authenticated in Administration Console or Test Platform) can **see and run all transactions / sequences, no matter what their accessibility is**.
- The **test user** (authenticated in the Test Platform or in case of anonymous access) can **see and run public transactions/sequences** and **run hidden ones if he knows their execution URL**.



# 12.1 Accessibility property

For a **transaction**,  
the **Accessibility property** is **private by default**.

For a **sequence**,  
the **Accessibility property** is **public by default**.

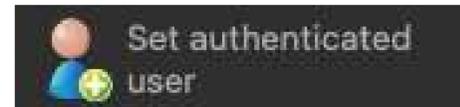


Property	Value
Base properties	
Accessibility	Private
Comment	
HTTP verb	GET
Response timeout	60
Sub path	search/movie?query={movieTitle}&in

Property	Value
Base properties	
Accessibility	Public
Comment	
Response timeout	60



## 12.2 Http session management steps



### Set authenticated user

This step sets a **user ID** as the **authenticated user ID** in the **current context/session** and sets the **current context/session** as **authenticated**.



### Remove authenticated user

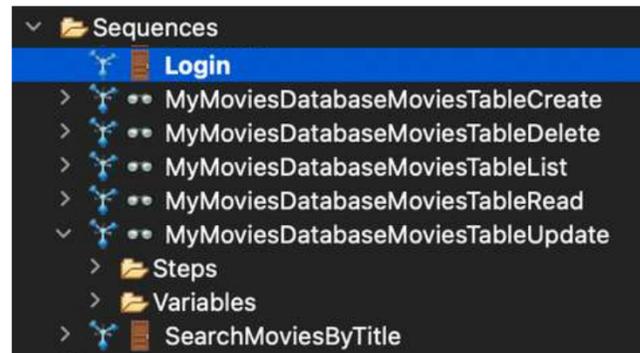
This step allows to easily **remove a value stored in the session** using its **key**, i.e. the **variable name**.



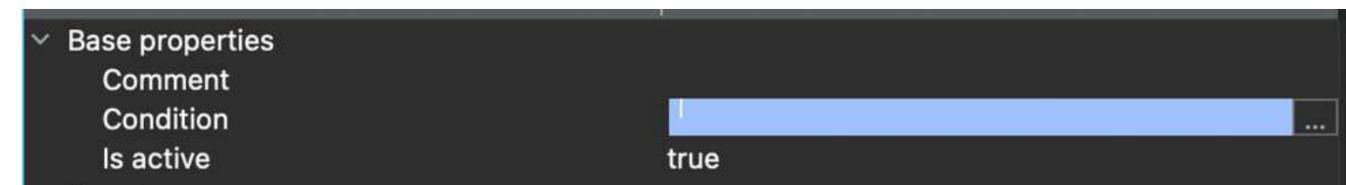
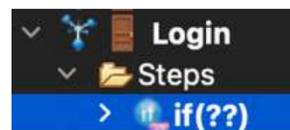
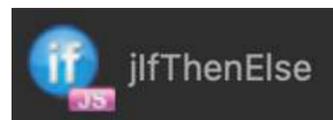
# 12.3 Create a Login sequence

Let's create a basic authentication in the project. We don't have a database with user accounts to log in so we will **simulate a user with a fake email and password** with hardcoded values.

Add a new sequence and rename it **Login**.



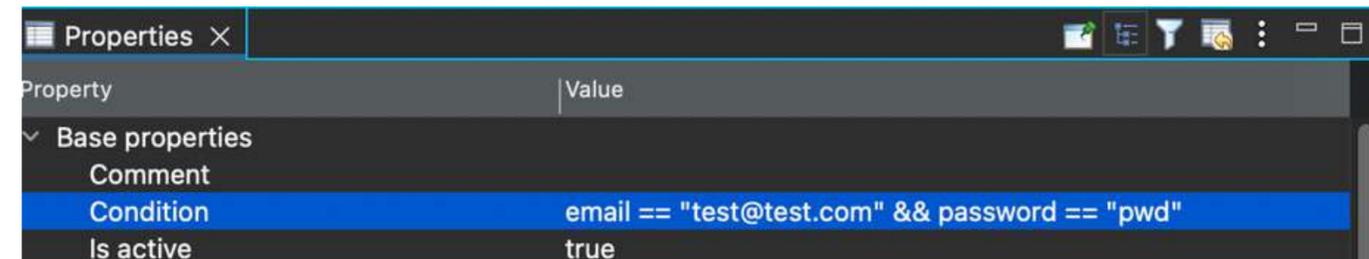
Add a **jIfThenElse** step in the Login sequence.



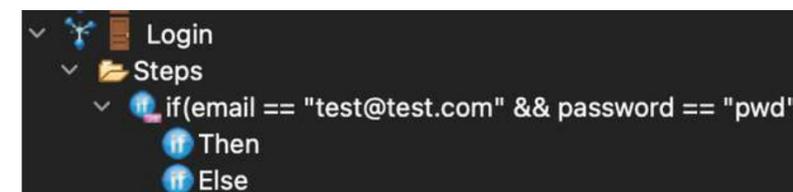
In the **Condition** property of the **jIfThenElse** step, let's add a condition with an email and a password.



```
email == "test@test.com" && password == "pwd"
```



It appears like this in the treeview.



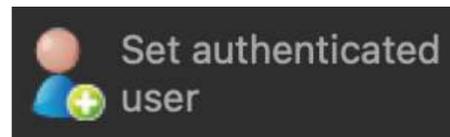
# 12.3 Create a Login sequence

```

Login
├── Steps
│   └── if(email == "test@test.com" && password == "pwd")
│       ├── Then
│       └── Else
    
```



In the Then of the jIfThenElse step, add a **Set authenticated user** step.



```

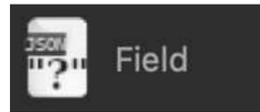
Login
├── Steps
│   └── if(email == "test@test.com" && password == "pwd")
│       ├── Then
│       │   └── authenticatedUserID
│       └── Else
    
```



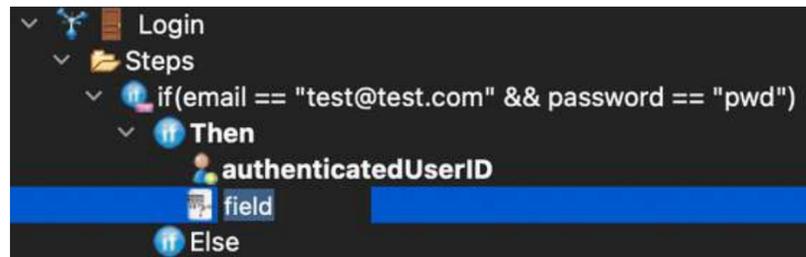
In the **User ID** property of the **Set authenticated user** step, enter **email** as value in **JS**.



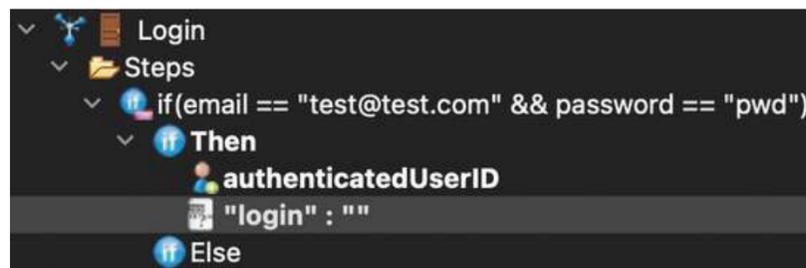
# 12.3 Create a Login sequence



Add a **Field** step after the **Set authenticated user** step



Rename it login.



In the **Value** property of the login field step

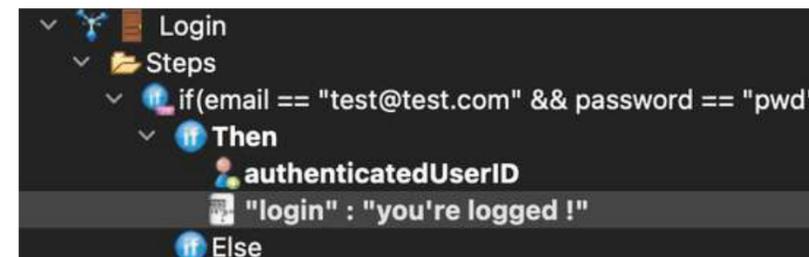
Property	Value
Comment	
Is active	true
Key	login
Type	string
Value	



Enter "You're logged !" (or any other message) as value in text (TX)



Property	Value
Comment	
Is active	true
Key	login
Type	string
Value	you're logged !

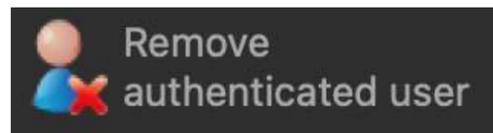


# 12.3 Create a Login sequence

```
Login
├── Steps
│   ├── if(email == "test@test.com" && password == "pwd")
│   │   ├── Then
│   │   │   ├── authenticatedUserID
│   │   │   └── "login" : "you're logged !"
│   │   └── Else
```



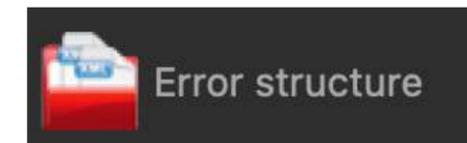
In the **Else** of the **jIfThenElse** step, add a **Remove authenticated user** step.



```
Login
├── Steps
│   ├── if(email == "test@test.com" && password == "pwd")
│   │   ├── Then
│   │   │   ├── authenticatedUserID
│   │   │   └── "login" : "you're logged !"
│   │   └── Else
│   │       └── Remove_authenticated_user
```



In the **Else** of the **jIfThenElse** step, after the **Remove authenticated user** step, add an **Error structure** step.



```
Login
├── Steps
│   ├── if(email == "test@test.com" && password == "pwd")
│   │   ├── Then
│   │   │   ├── authenticatedUserID
│   │   │   └── "login" : "you're logged !"
│   │   └── Else
│   │       ├── Remove_authenticated_user
│   │       └── Error_structure
```



# 12.3 Create a Login sequence

```

Login
├── Steps
│   ├── if(email == "test@test.com" && password == "pwd")
│   │   ├── Then
│   │   │   ├── authenticatedUserID
│   │   │   └── "login" : "you're logged !"
│   │   └── Else
│   │       └── Remove_authenticated_user
│   └── Error_structure
    
```



In the **Error structure step properties**

Property	Value
Code	
Comment	
Details	
Is active	true
Message	



In the **Message property**,  
add **Problem with login** as value.

Property	Value
Code	
Comment	
Details	
Is active	true
Message	Problem with login

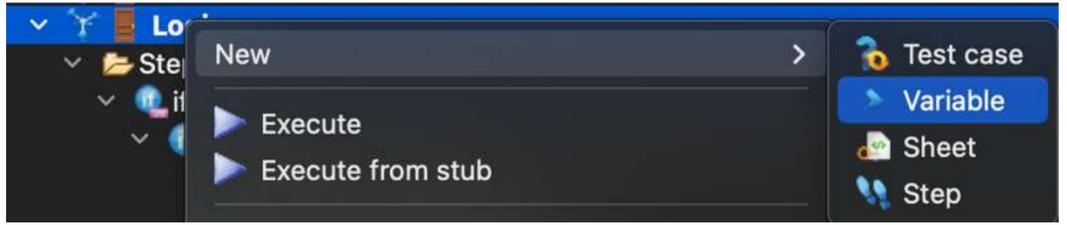
```

Login
├── Steps
│   ├── if(email == "test@test.com" && password == "pwd")
│   │   ├── Then
│   │   │   ├── authenticatedUserID
│   │   │   └── "login" : "you're logged !"
│   │   └── Else
│   │       └── Remove_authenticated_user
│   └── <error> [] Problem with login
    
```

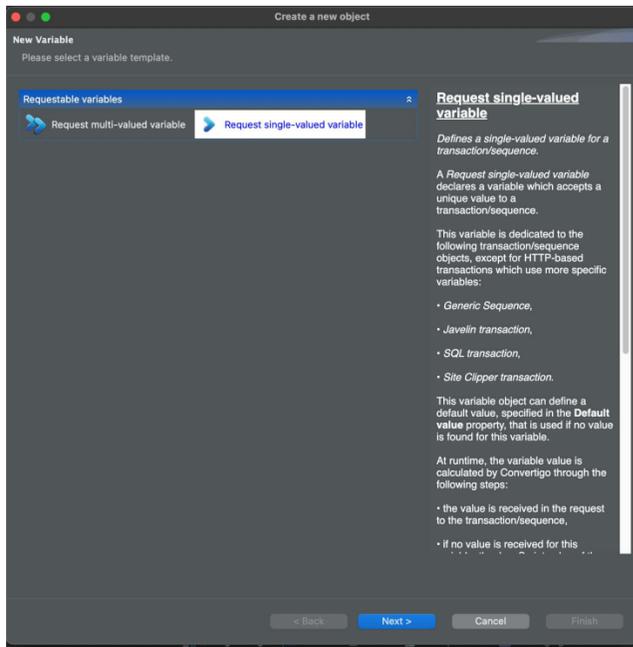


# 12.3 Create a Login sequence

Let's add email and password variables to the Login sequence.



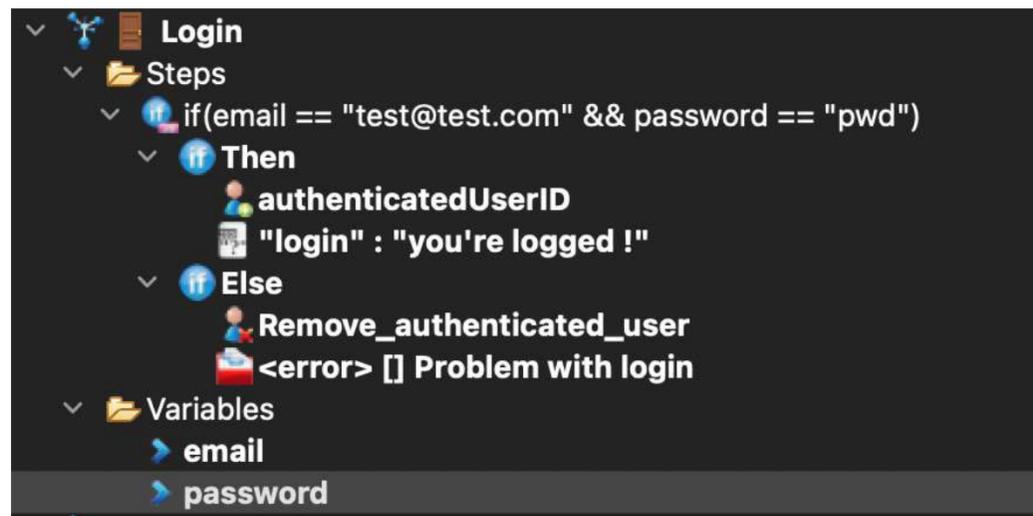
↓



➤ Request single-valued variable



Our Login sequence with error management is completed



```

Login
├── Steps
│   ├── if(email == "test@test.com" && password == "pwd")
│   │   ├── Then
│   │   │   ├── authenticatedUserID
│   │   │   └── "login" : "you're logged !"
│   │   └── Else
│   │       ├── Remove_authenticated_user
│   │       └── <error> [] Problem with login
│   └── Variables
│       ├── email
│       └── password
└── Variables
    ├── email
    └── password
    
```

➤ Variables

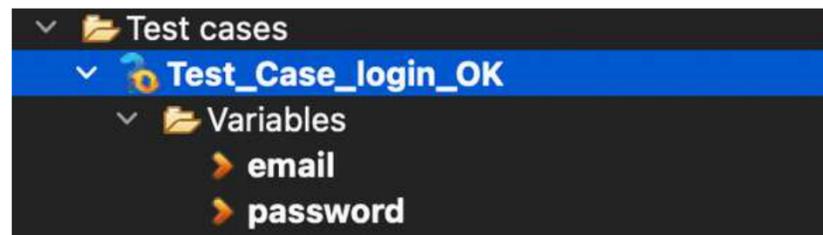
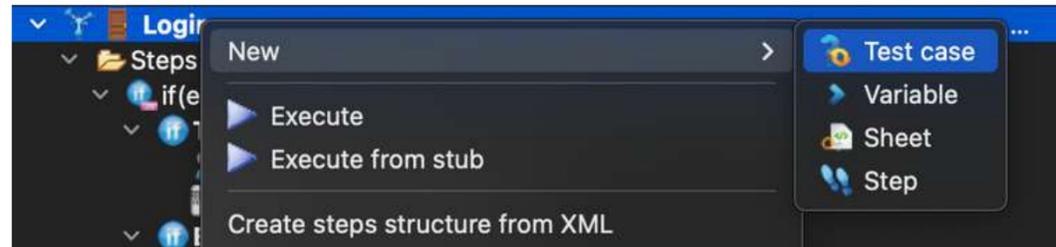
- email
- password



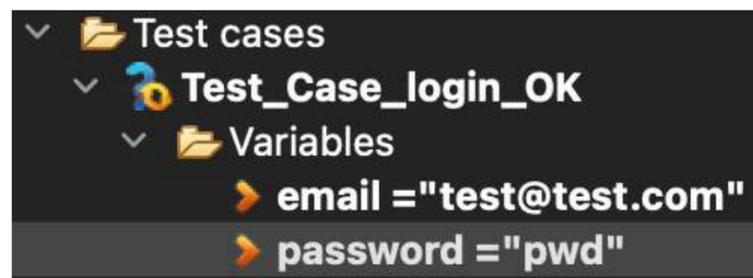
# 12.4 Test the Login sequence

Now, let's create a test case with the right email and password values.

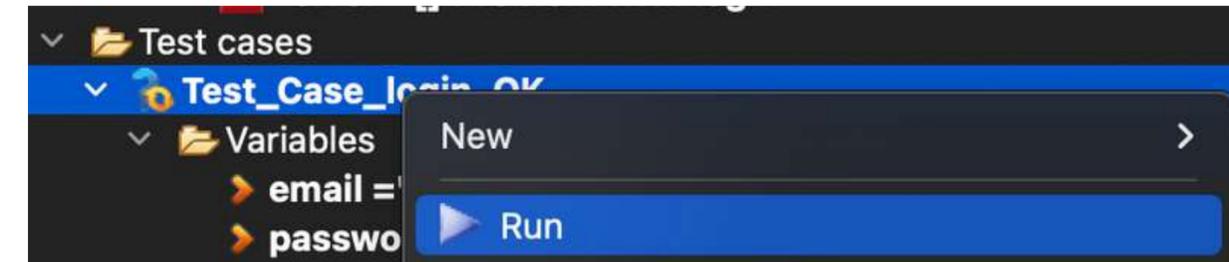
```
if(email == "test@test.com" && password == "pwd")
```



Enter the right email and password values in the test case corresponding variables



Run the test case.



The sequence returns the login message.



# 12.4 Test the Login sequence

Now, let's create a test case with the right email and a wrong password

```
Test_Case_wrong_password
├── Variables
│   ├── email = "test@test.com"
│   └── password = "wrong password"
```



Run the test case.

```
Test_Case_wrong_password
├── Variables
│   ├── email = "test@test.com"
│   └── password = "wrong p
```

New

Run



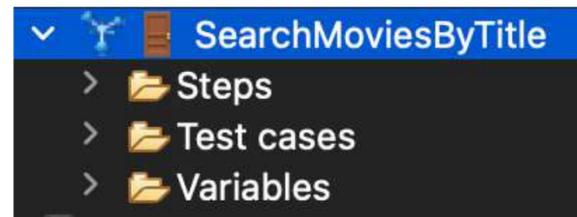
The sequence returns an error with the error message we indicated.

```
MyMoviesProject [S: Login].json
1 {
2   "error": {
3     "code": "-1",
4     "message": "Problem with login",
5     "details": "",
6     "context": "",
7     "exception": "",
8     "stacktrace": "",
9   "attr": {
10    "project": "MyMoviesProject",
11    "sequence": "Login",
12    "type": "project"
13  }
14 }
15 }
```

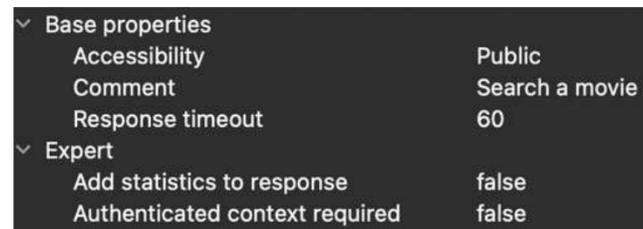


# 12.5 Add authentication to a sequence

Let's say we want to add authentication to the SearchMoviesByTitle sequence.



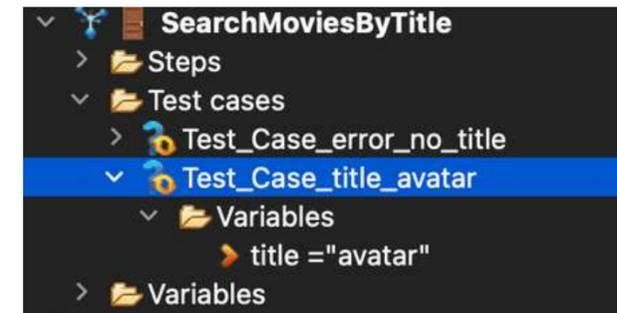
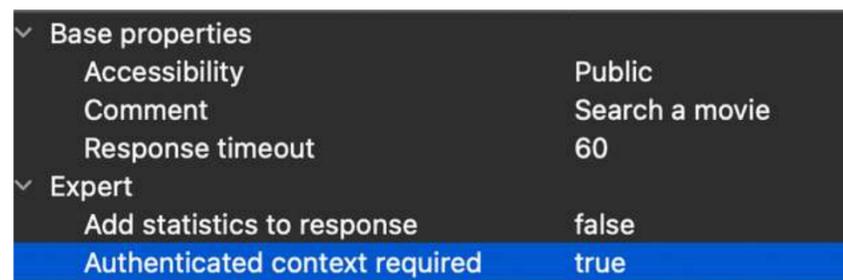
In the **sequence properties**



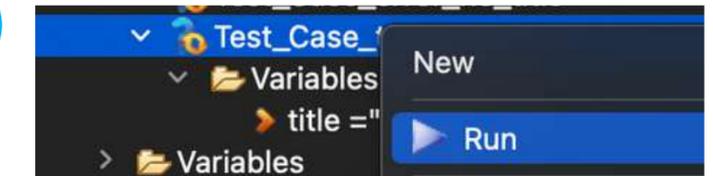
Let's change the value of the



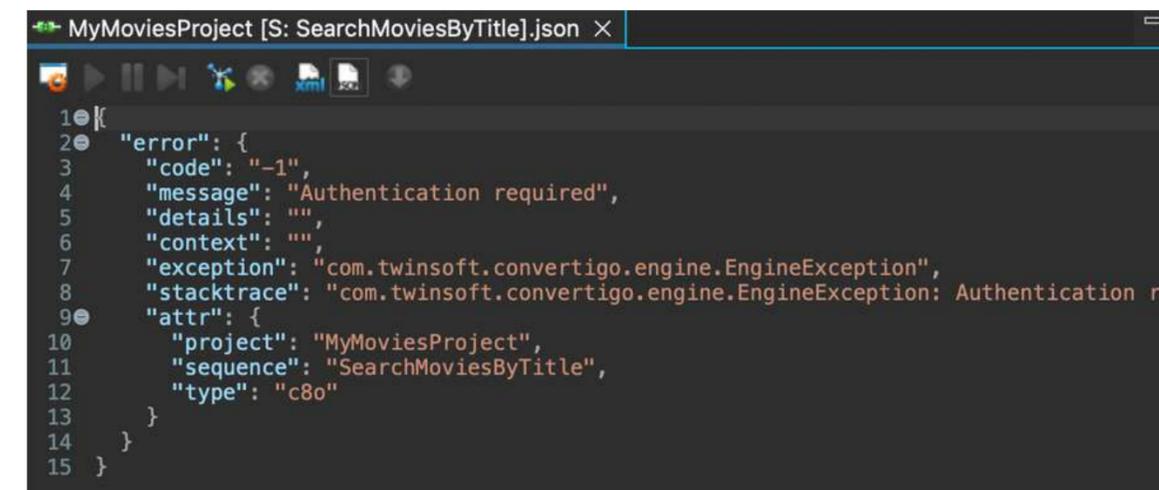
**Authenticated context required property** from **false** to **true**.



Run the test case.

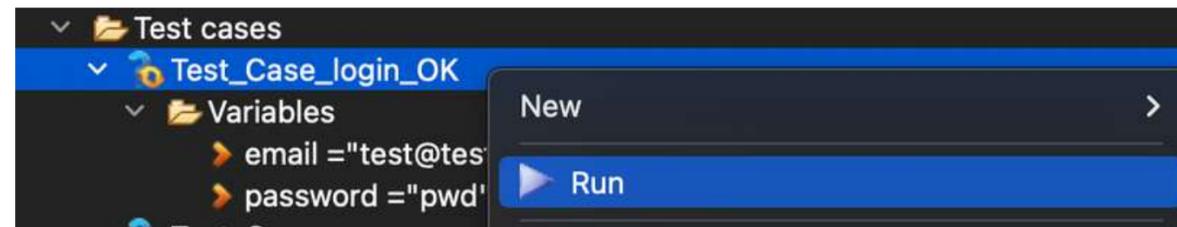


The sequence returns an error with an error message : **Authentication required.**

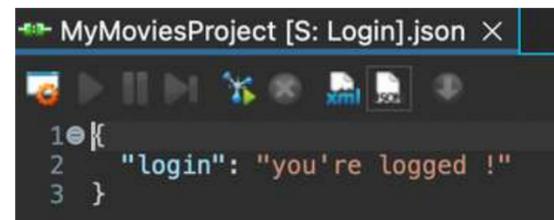


# 12.5 Add authentication to a sequence

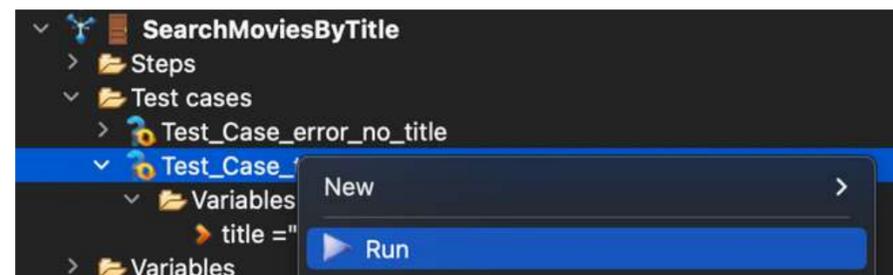
Now, let's run the Login sequence test case before the SearchMoviesByTitle sequence test case.



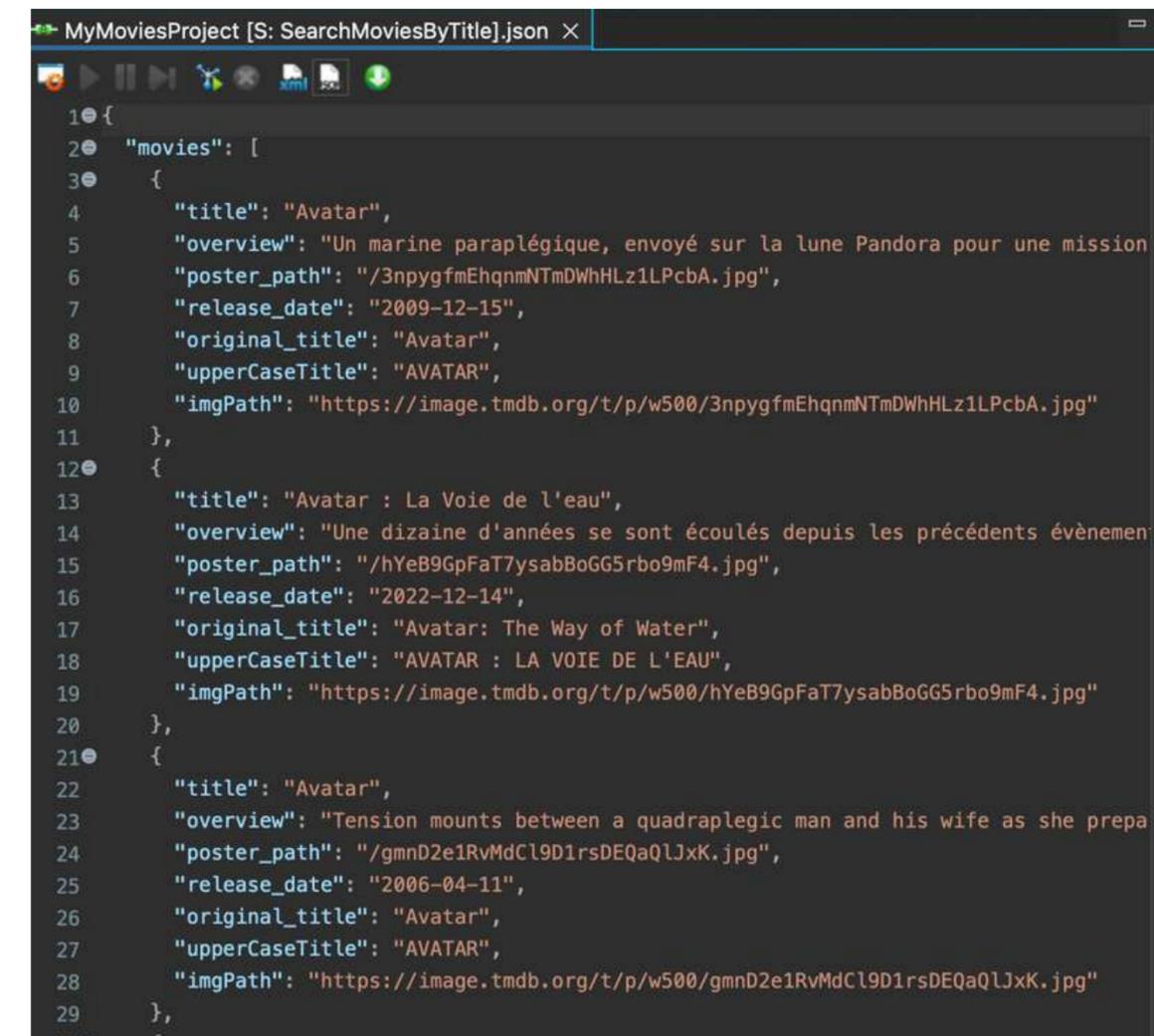
The sequence returns the login message and we are logged in.



Run the SearchMoviesByTitle sequence test case again.



The sequence returns data from the TMDb API.



# Appendix



**A.1** Use Java classes in JS SCOPE

---

**A.2** Generate documentation in ReadMe file

# A.1 Use Java classes in JS SCOPE

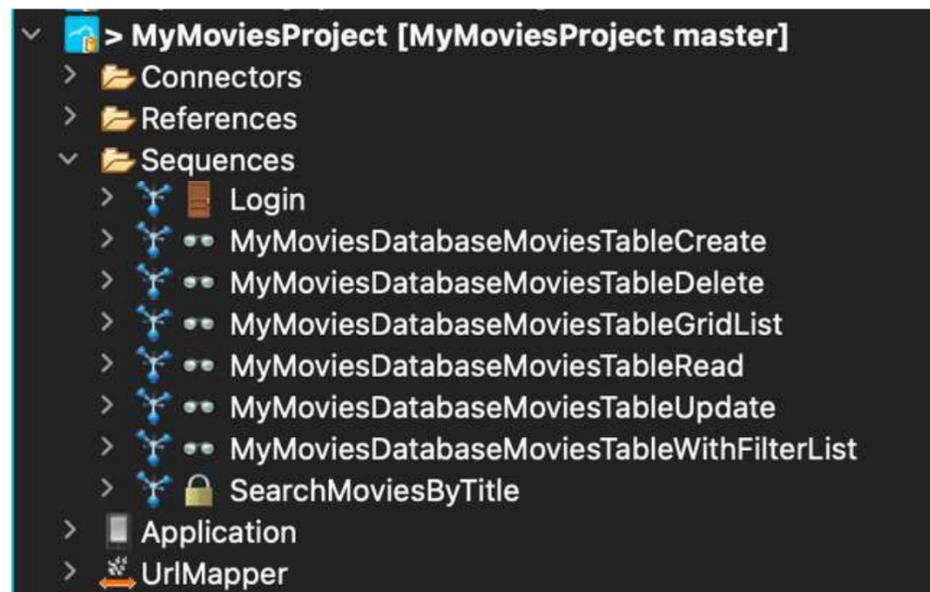
In the studio, we can use Java classes in the JS Scope (thanks to the Rhino js engine).

We can integrate **calls to Java code from the studio or external libraries into the sequence** in a **Sequence JS step**.

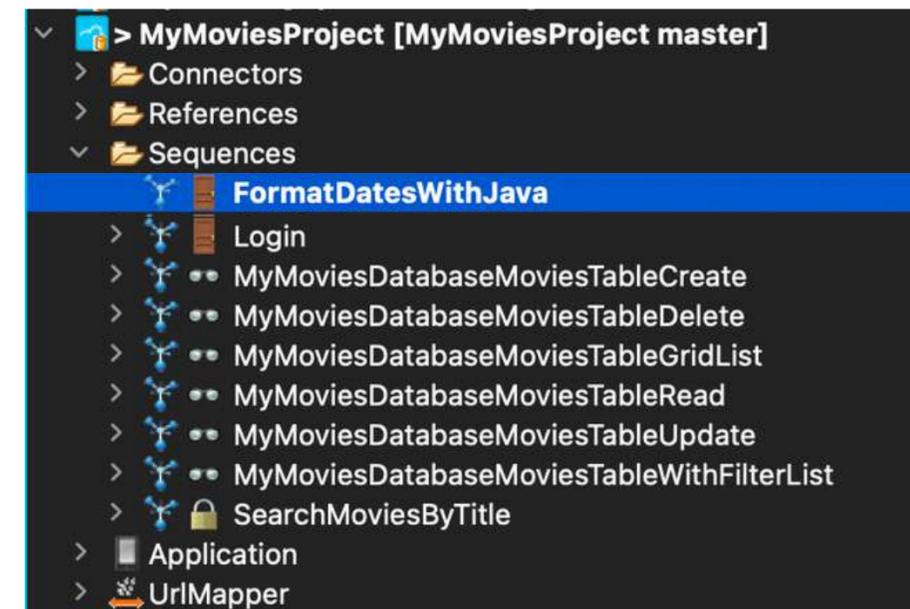
Let's say we want to **manage dates with Java classes** in our application.

We can use

- the **Calendar class** for handling date and time in Java.
- the **SimpleDateFormat class** to format Calendar instances in a human-readable way.

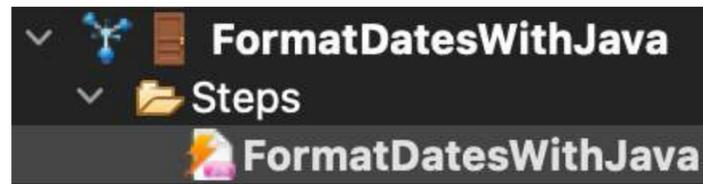


Add a new sequence  
and name it **FormatDatesWithJava**



# A.1 Use Java classes in JS SCOPE

Add a Sequence JS step in the sequence, and name it FormatDatesWithJava.



Click twice on the **Sequence JS step** to open the **JS file** in the editor panel



Edit the JS file as following.

```
FormatDatesWithJava.js ×
1 // Import the Java Classes with JavaImporter
2 var JI = JavaImporter(
3     Packages.java.util.Calendar,
4     Packages.java.text.SimpleDateFormat
5 );
6
7 //Declare a variable
8 var formattedDate;
9
10 with (JI) {
11
12     // Create a Calendar instance
13     let calendar = Calendar.getInstance();
14
15     // Create a SimpleDateFormat instance with desired format
16     let dateFormat = new SimpleDateFormat("EEEE d MMMM yyyy");
17
18     // Format the calendar's time
19     formattedDate = dateFormat.format(calendar.getTime());
20 }
21
```



# A.1 Use Java classes in JS SCOPE

```
FormatDatesWithJava.js ×
1 // Import the Java Classes with JavaImporter
2 var JI = JavaImporter(
3     Packages.java.util.Calendar,
4     Packages.java.text.SimpleDateFormat
5 );
6
7 //Declare a variable
8 var formattedDate;
9
10 with (JI) {
11     // Create a Calendar instance
12     let calendar = Calendar.getInstance();
13
14     // Create a SimpleDateFormat instance with desired format
15     let dateFormat = new SimpleDateFormat("EEEE d MMMM yyyy");
16
17     // Format the calendar's time
18     formattedDate = dateFormat.format(calendar.getTime());
19 }
20
21
```

Here, a Calendar instance is created and The SimpleDateFormat is used to format this date into a readable format.

In the pattern "EEEE d MMMM yyyy", we have

- the day of the week (EEEE)
- the day of the month (d)
- the month in full (MMMM)
- and the year (yyyy)

This will produce output like "Mercredi 15 février 2023".

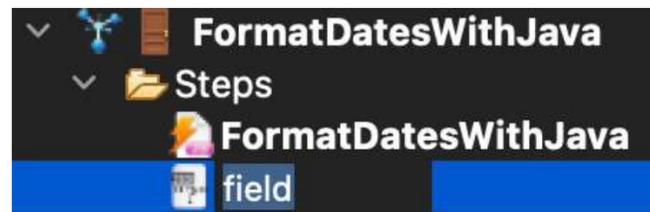
You can adjust the pattern in SimpleDateFormat to suit your preferred date format.



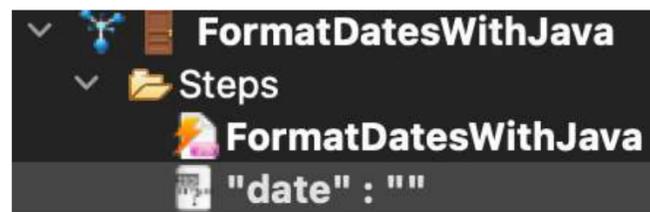
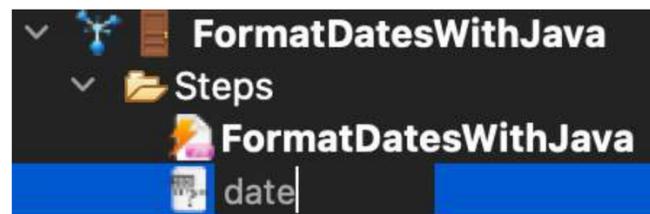
# A.1 Use Java classes in JS SCOPE

## Add a Field step

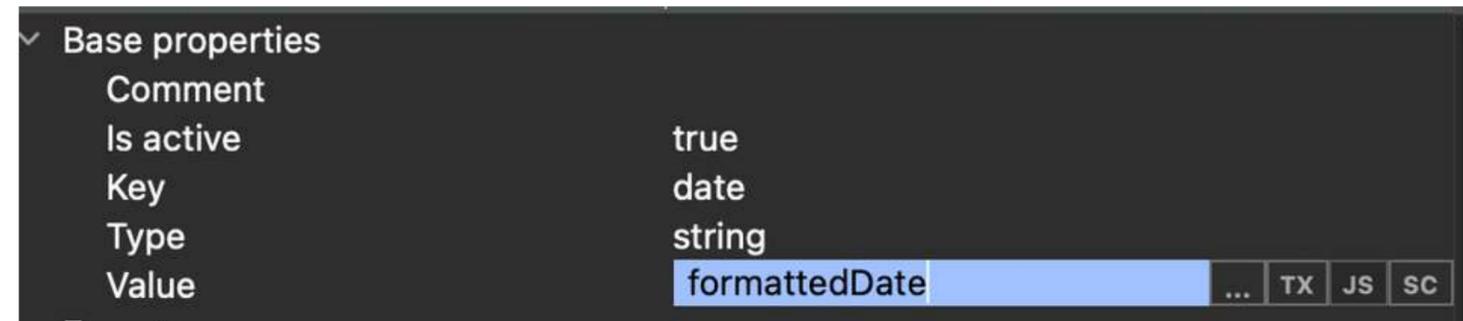
after the Sequence JS step.



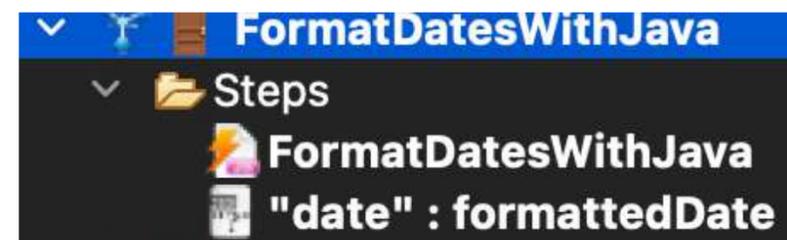
Name it **date**.



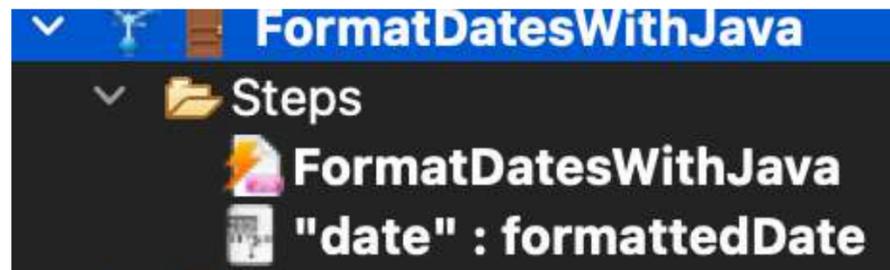
in the **Value** property of the Field step properties, enter **formattedDate** as value in JS.



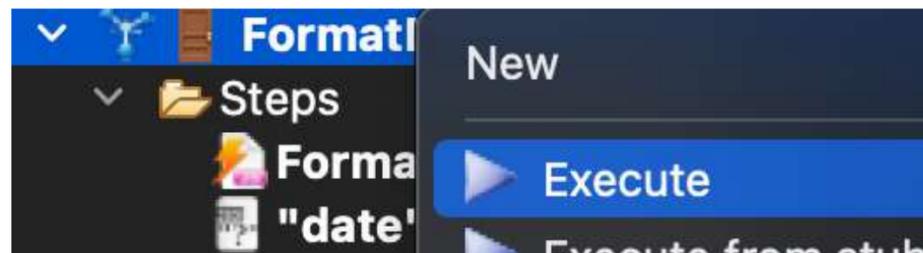
**Value** =formattedDate



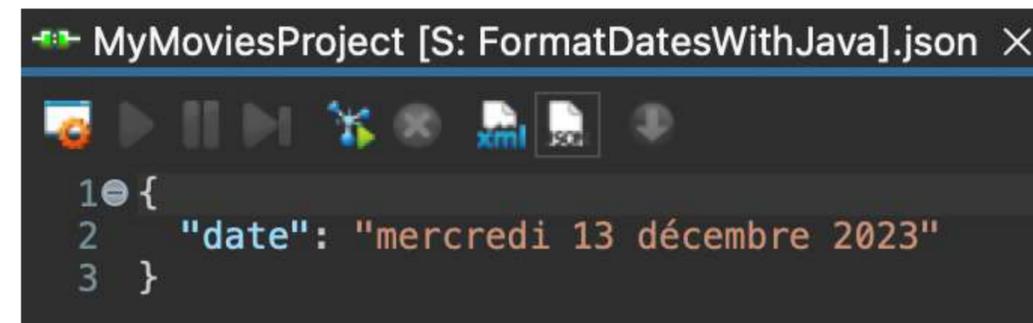
# A.1 Use Java classes in JS SCOPE



Execute the Sequence.



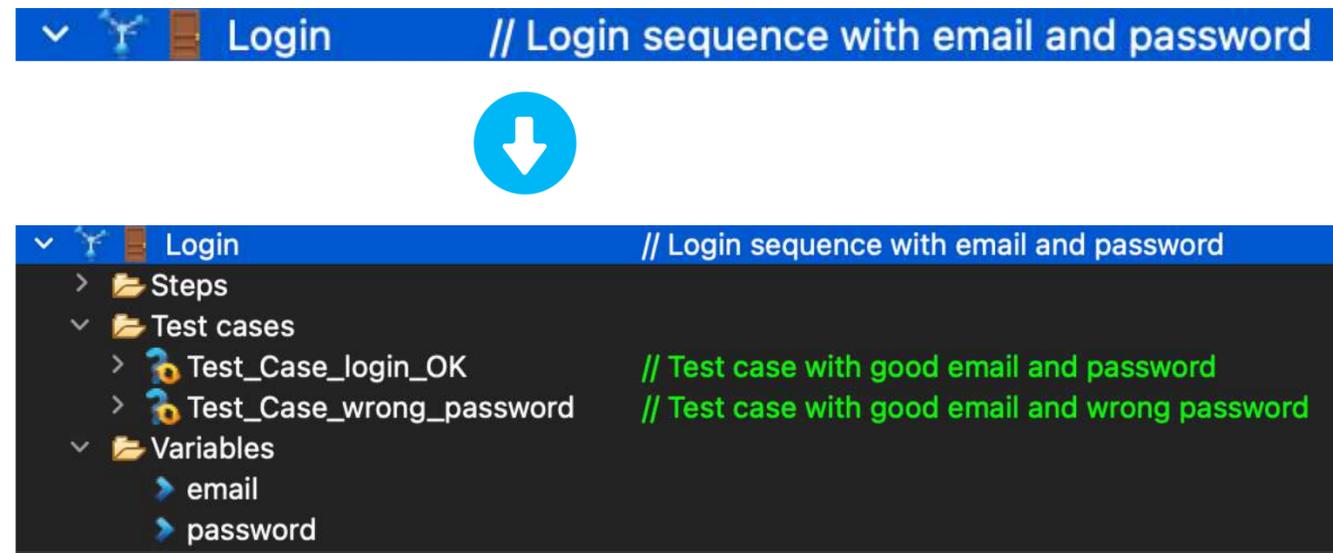
The sequence returns the formatted date.



# A.2 Generate documentation in ReadMe file

In the studio, we can automatically generate a documentation in the ReadMe.md file.

You can edit the comment in the treeview.

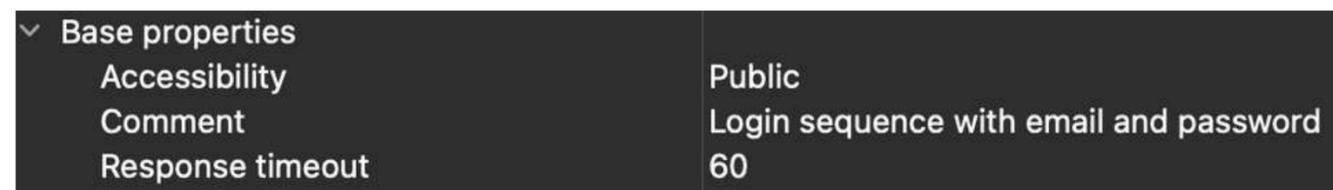


▼ Login // Login sequence with email and password

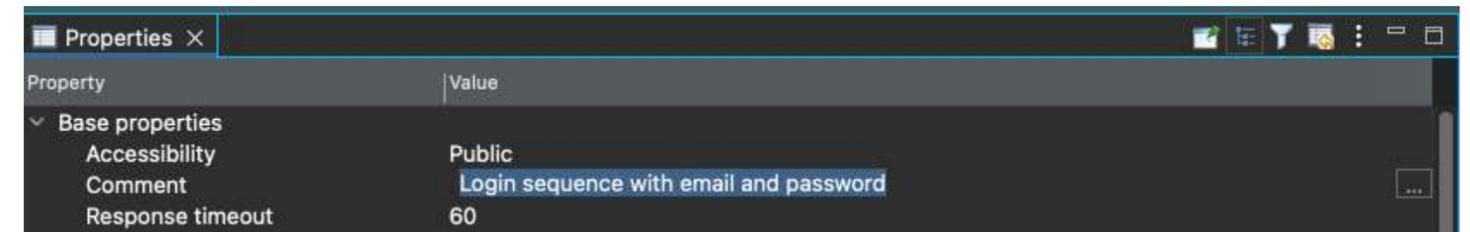
▼ Login // Login sequence with email and password

- > Steps
- ▼ Test cases
  - > Test\_Case\_login\_OK // Test case with good email and password
  - > Test\_Case\_wrong\_password // Test case with good email and wrong password
- ▼ Variables
  - > email
  - > password

Or in the comment property.

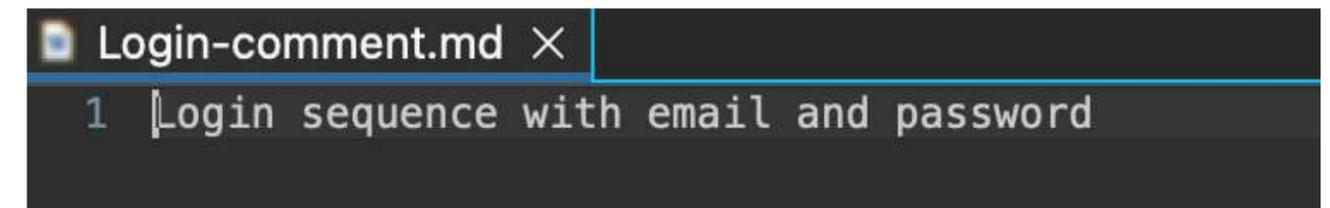
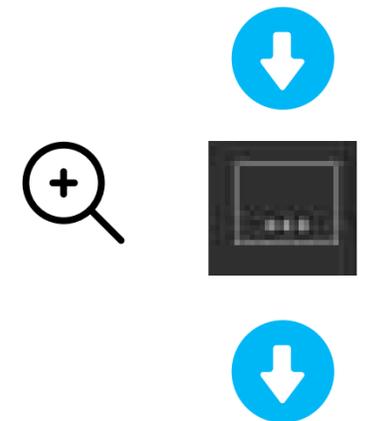


Property	Value
Base properties	
Accessibility	Public
Comment	Login sequence with email and password
Response timeout	60



Property	Value
Base properties	
Accessibility	Public
Comment	Login sequence with email and password
Response timeout	60

You can also click on this icon to open the markdown file of the comment and write a more detailed comment, if needed.

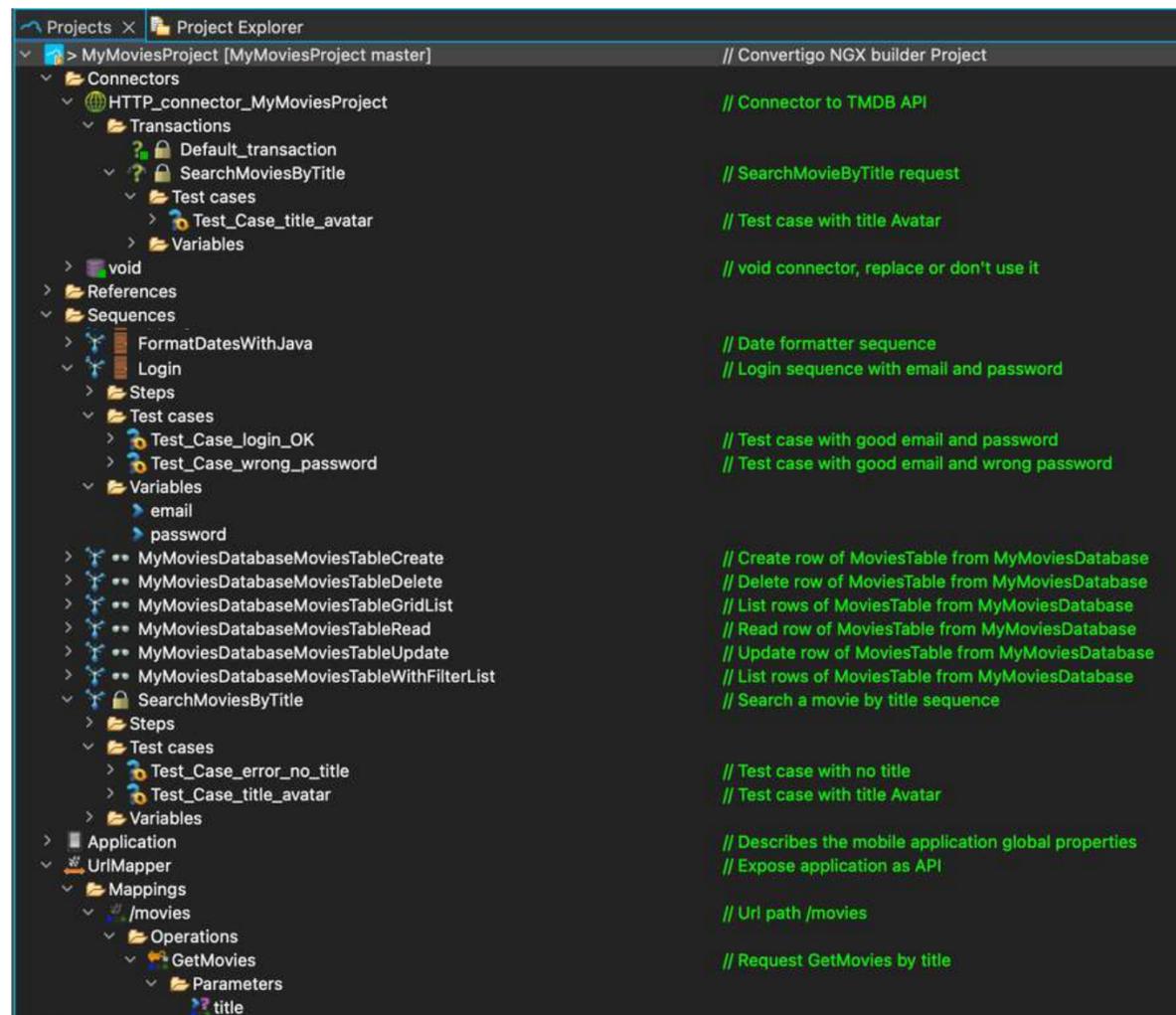


```
Login-comment.md X
1 | Login sequence with email and password
```

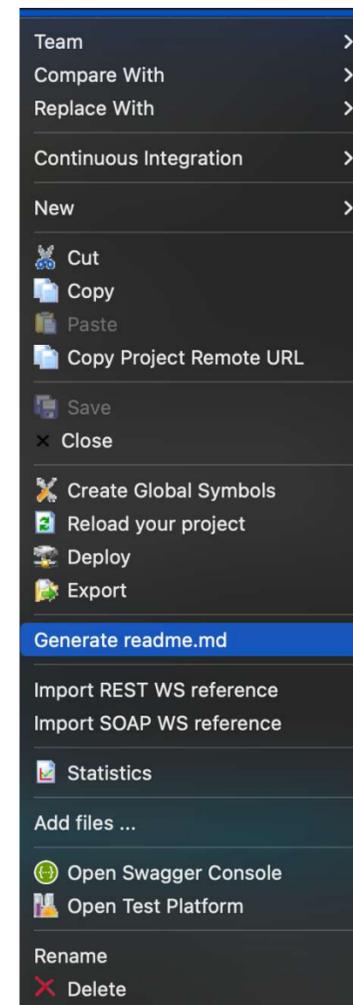


# A.2 Generate documentation in ReadMe

You can write comments on every step in the treeview.



MyMoviesProject [MyMoviesProject master]



To enable the automatic generation of the documentation in the ReadMe.md file, right-click on the project name in the Projects view.



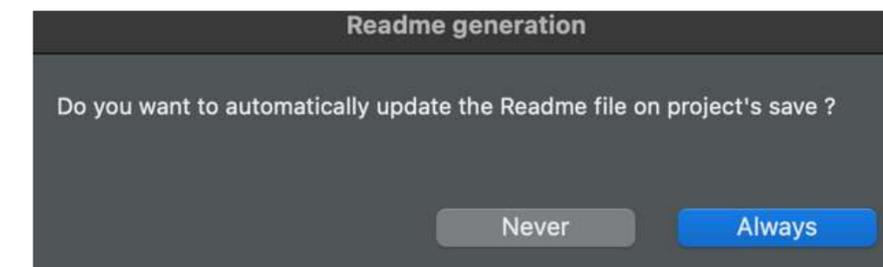
Select **Generate readme.md** to open the Readme generation window.



Generate readme.md



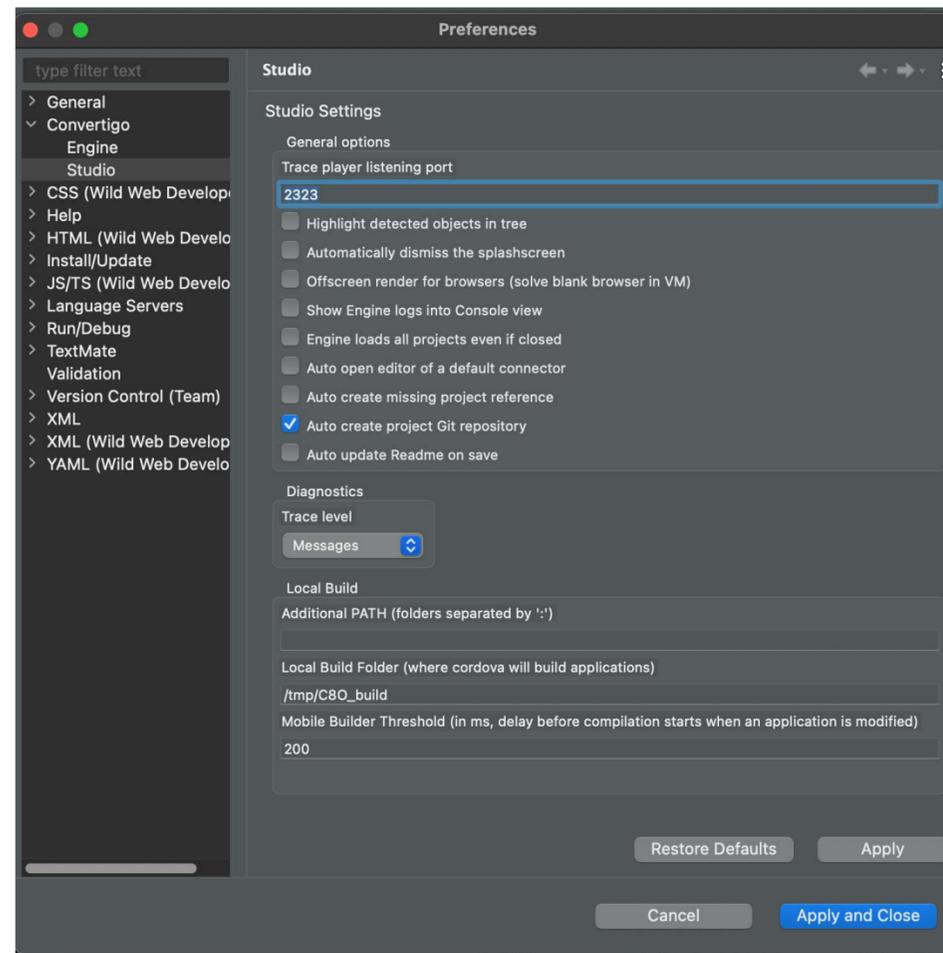
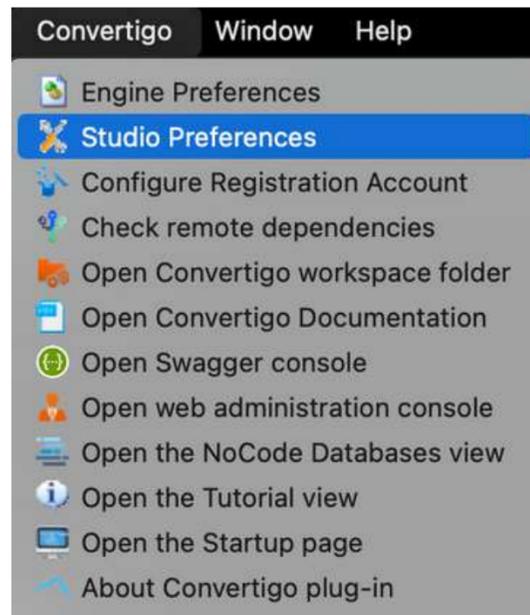
Click on **Always** to confirm.



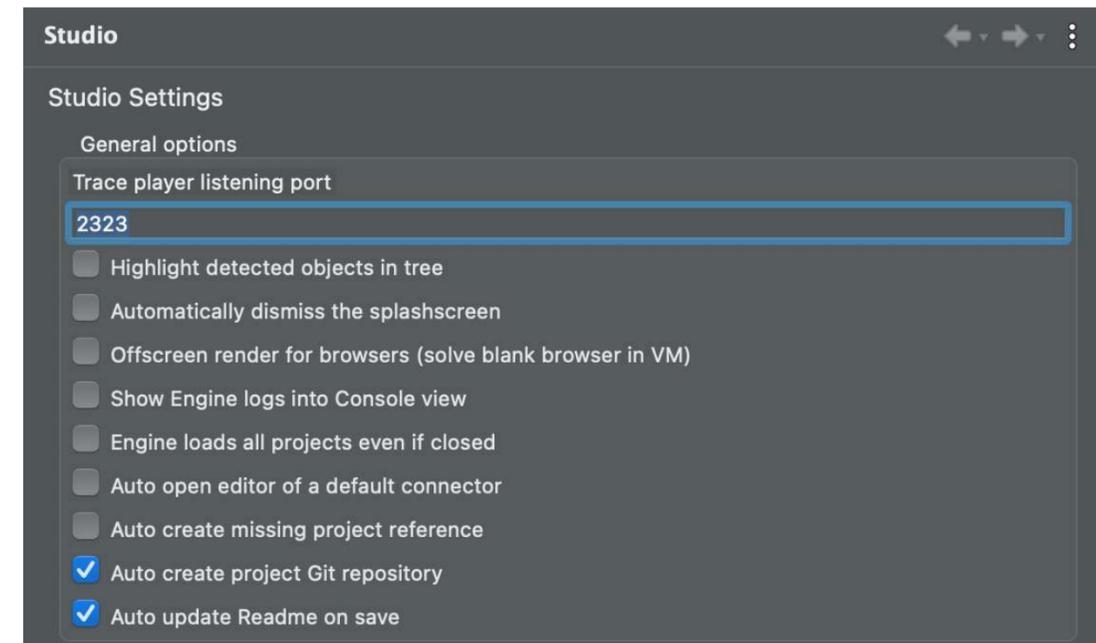
# A.2 Generate documentation in ReadMe

You can also use **Studio Preferences** to enable the automatic generation of the documentation in the ReadMe.md file. Each time the project is saved, the file is updated.

In the Preferences window



Click on **Auto update Readme on save**.



Auto update Readme on save



Click on **Apply and Close**.

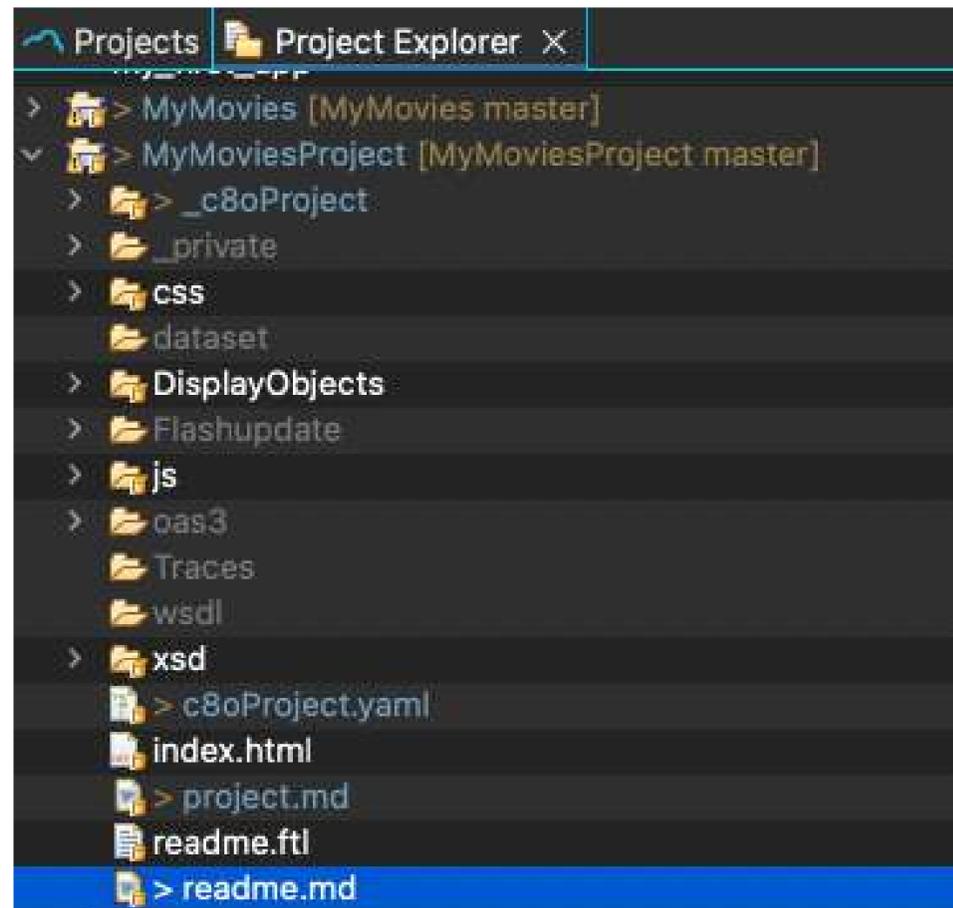


Apply and Close

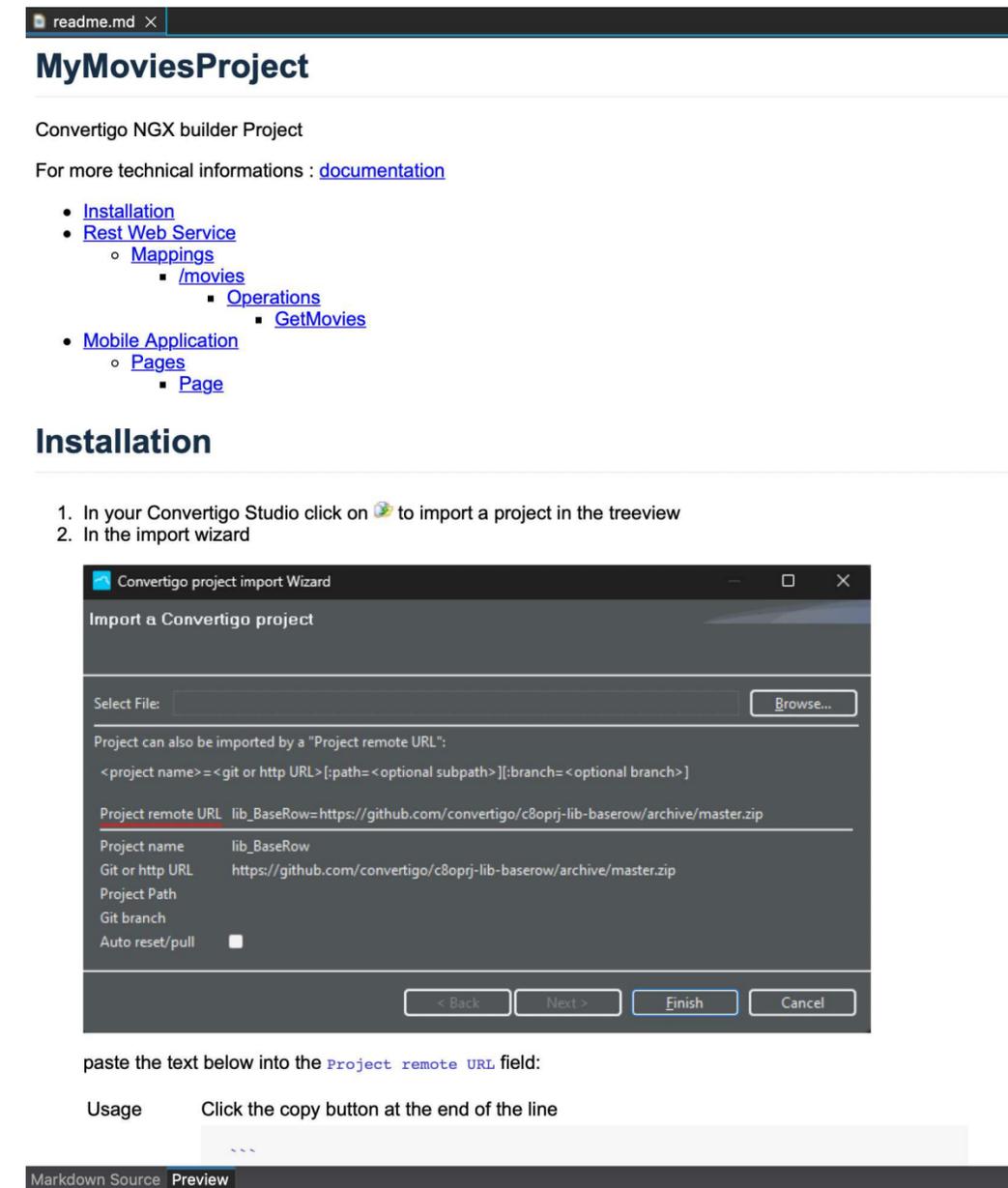


# A.2 Generate documentation in ReadMe

In the **Project Explorer** view, click twice on the **readme.md** file to open it.



The content of the file is automatically generated



**MyMoviesProject**

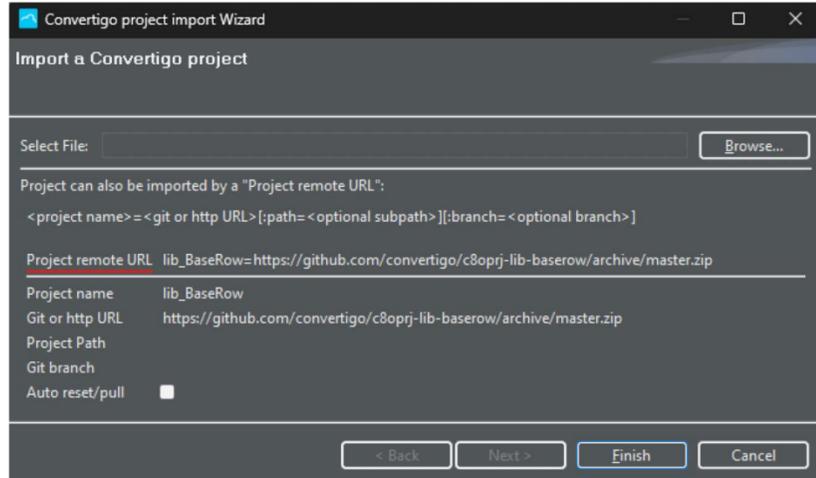
Convertigo NGX builder Project

For more technical informations : [documentation](#)

- [Installation](#)
- [Rest Web Service](#)
  - [Mappings](#)
    - [/movies](#)
      - [Operations](#)
      - [GetMovies](#)
- [Mobile Application](#)
  - [Pages](#)
    - [Page](#)

### Installation

1. In your Convertigo Studio click on  to import a project in the treeview
2. In the import wizard



paste the text below into the **Project remote URL** field:

Usage      Click the copy button at the end of the line

```
***
```

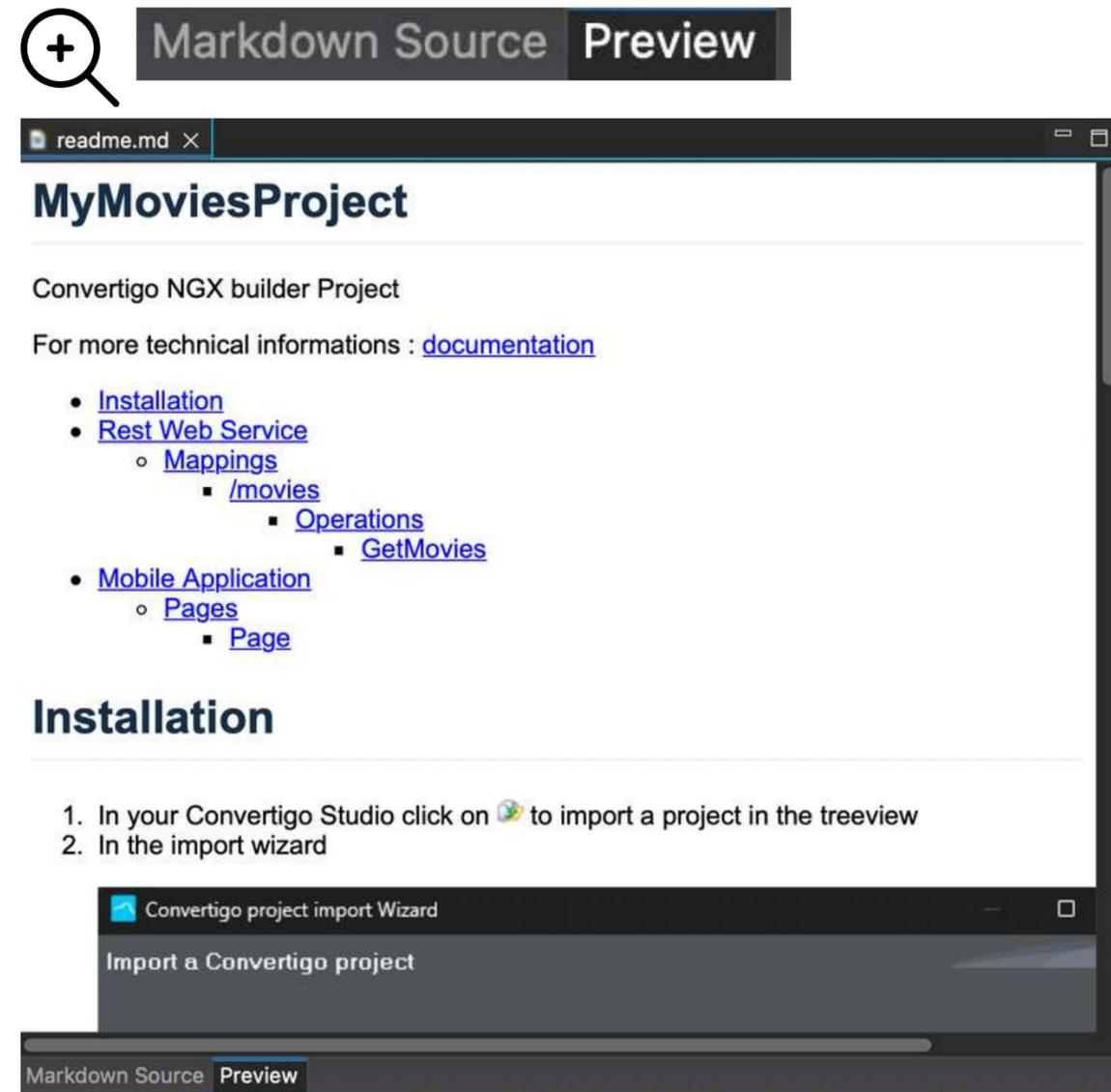
Markdown Source    Preview



# A.2 Generate documentation in ReadMe

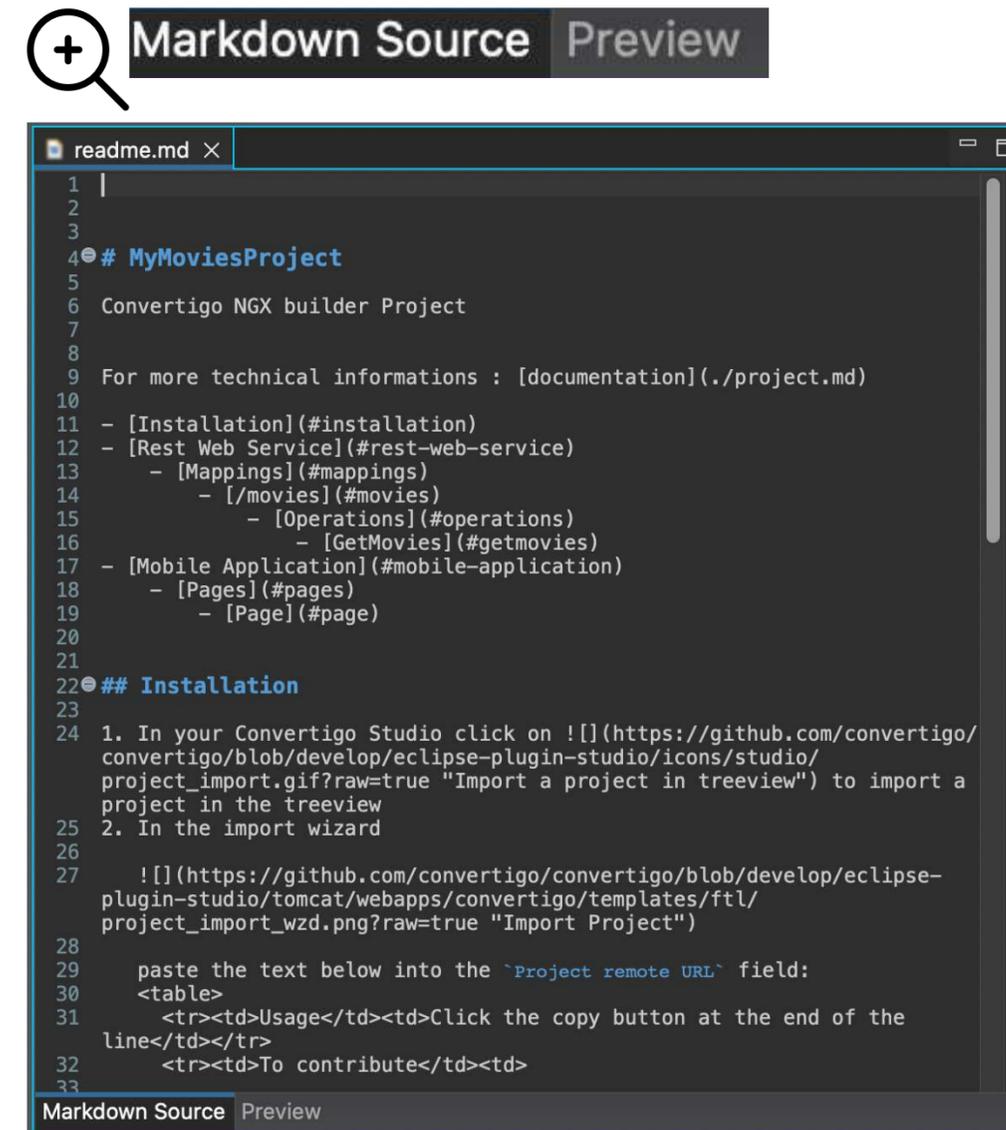
The ReadMe.md file is **displayed in Preview by default**. You can switch it to **Markdown** to **edit it in markdown**.

Preview view



A screenshot of a web browser showing the preview of a ReadMe file. At the top, there are two tabs: 'Markdown Source' and 'Preview', with 'Preview' selected. The main content area displays the rendered HTML of the ReadMe file, including a title 'MyMoviesProject', a subtitle 'Convertigo NGX builder Project', a link to 'documentation', a list of links for 'Installation', 'Rest Web Service', and 'Mobile Application', and an 'Installation' section with a numbered list of steps. A small 'Convertigo project import Wizard' dialog box is overlaid on the bottom part of the page.

Markdown Source view



A screenshot of a web browser showing the source code of the ReadMe file. At the top, there are two tabs: 'Markdown Source' and 'Preview', with 'Markdown Source' selected. The main content area displays the raw markdown text, including headers, lists, and code blocks. The code is line-numbered from 1 to 33.

