Guide Utilisateur – Formation ABAP RAP

Création d'applications avec le modèle de programmation ABAP Restful

2025

Introduction

Ce guide pratique a été conçu pour documenter toutes les étapes de la formation

"**Building Apps with the ABAP RESTful Application Programming Model**" (**RAP**).

Il présente, de manière structurée, les concepts clés, les outils utilisés, les étapes de développement ainsi que les bonnes pratiques à suivre dans un environnement SAP moderne.

Chaque semaine de formation est détaillée avec ses unités correspondantes, allant de la configuration initiale à la consommation de services via API, afin de fournir une vision complète du cycle de vie d'une application RAP.

Table des matières

1. Semaine 0 – Préparation de l'environnement de développement.....5
1.1 Prérequis techniques.....5
1.2 Étapes de configuration dans Eclipse.....6
1.2.1 Installation d'Eclipse et des outils ABAP.....6
1.2.2 Installation du plugin ABAP Development Tools (ADT) 7
1.2.3 Configuration d'ADT pour l'utilisation avec SAP Build (Link Handler) 10
1.3 Création d'un projet ABAP Cloud via SAP Build.....15

2. Semaine 1 - Introduction – Modèle de programmation d'applications ABAP RESTful..28
2.1 Vue d'ensemble (Unité 1) 28
2.1.1 Qu'est-ce que RAP ?.....28
2.1.2 Pourquoi RAP a été introduit ?.....28
2.1.3 Composants clés du modèle RAP.....29

2.2 Présentation de l'architecture (Unité 2) 29
2.2.1 Vue d'ensemble de l'architecture RAP.....29
2.2.2 Description des couches principales.....30
2.2.3 Cycle de vie des données dans RAP.....31

2.3 Technologies impliquées (Unité 3) 31
2.3.1 Technologies utilisées dans RAP.....31
2.3.2 Synergie des composants.....32

2.4 Présentation de SAP Cloud Platform, environnement ABAP (Unité 4) 32
2.4.1 Qu'est-ce que SAP BTP – ABAP Environment ?.....32
2.4.2 Fonctionnalités offertes.....32

2.5 Création de votre première application console ABAP Cloud (Unité 5) 33
2.5.1 Création d'un package ABAP.....33
2.5.2 Création d'une classe ABAP Hello World.....36

3. Semaine 2 – Développement d'une application de rapport de liste en lecture seule.....42
3.1 Le scénario commercial (Unité 1) 42
3.2 Création des Tables de Base de Données (Unité 2) 43
3.3 Création du modèle de données CDS (Core Data Service) (Unité 3) 58
3.4 Création d'une projection de modèle de données CDS (Unité 4) 67
3.5 Enrichissement de modèle de données CDS
3.6 Enrichissement de modèle de données CDS projeté avec des métadonnées UI (Unité 5) 77
3.7 Développement et prévisualisation du service commercial (OData) (Unité 6) 89
3.8 Mise en œuvre des autorisations de base dans RAP (Unité 7) 100

4. Semaine 3 — Activation du comportement transactionnel d'une application.....124
4.1 Définition du comportement basique de l'objet métier (Unité 1) 125
4.2 Création de la projection de comportement de l'objet métier (Unité 2) 138
4.3 Compréhension de langage de manipulation d'entités (EML) (Unité 3) 144
4.4 Amélioration du comportement des objets métier avec une logique spécifique à l'application (Unité 4) 162
4.5 MISE EN ŒUVRE DU COMPORTEMENT DE L'OBJET MÉTIER (Unité 5) 167
4.6 Activation de la gestion du projet (Unité 6) 218

5. Semaine 4 — Intégration de Code Métier Existant (Unmanaged Runtime) 238
5.1 Création du modèle de données CDS (Unité 1) 238
5.2 Définition et mise en œuvre du comportement des objets métier (Unité 2) 257
5.3 Création de la projection d'objet métier (Unité 3) 319
5.4 Création et prévisualisation du service d'interface utilisateur OData (Unité 4) 346

6. Semaine 5 – Consommation de services et API Web 354
6.1 Création du modèle de consommation de services (Unité 1) 354
6.2 Définition de l'entité CDS personnalisée (Unité 2) 375
6.3 Création de la classe d'implémentation (Unité 3) 382
6.4 Amélioration du modèle de vol avec des données externes (Unité 4) 387
6.5 Création et aperçu du service d'interface utilisateur OData (Unité 5) 398
6.6 Création et aperçu de l'API Web OData (Unité 6) 406

1. Semaine 0 – Préparation de l'environnement de développement

Avant de démarrer le développement avec le modèle **RAP** (ABAP RESTful Application Programming Model), il est nécessaire de configurer correctement l'environnement de travail. Cette section décrit étape par étape la configuration locale que j'ai utilisée, les outils installés, les limitations rencontrées, ainsi que les solutions apportées.

1.1.Prérequis techniques

Pour suivre la formation et développer une application **RAP**, voici les éléments nécessaires :

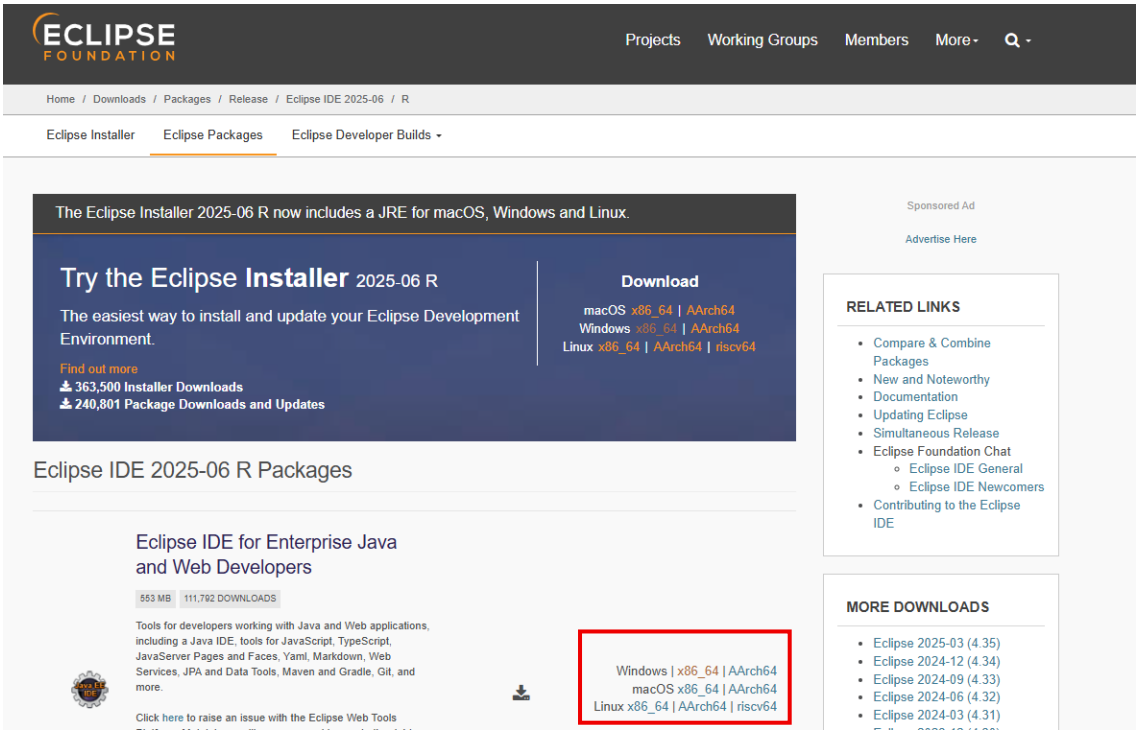
Élément	Description
SAP BTP (Business Technology Platform)	Plateforme cloud de SAP permettant de développer, tester et déployer des applications dans un environnement ABAP natif.
SAP Build	Environnement de développement visuel dans SAP BTP permettant de créer des applications Full-Stack ABAP, déclencher la création de projets ADT et gérer les packages.
Eclipse IDE + ABAP Development Tools (ADT)	Eclipse est un IDE open-source. Le plugin ADT permet de développer des objets ABAP (CDS, Behavior, Services, etc.) depuis un environnement local, connecté à SAP BTP.
Instance ABAP Cloud	Environnement de développement ABAP dans le cloud SAP BTP.
Navigateur moderne (Chrome, Edge...)	Requis pour accéder à la SAP Build Lobby, se connecter à SAP BTP et autoriser l'ouverture d'Eclipse via les liens eclipse+command://.
SAP GUI (optionnel)	Client SAP GUI, utilisé uniquement pour certaines vérifications techniques. Non nécessaire pour le développement RAP, qui se fait entièrement via Eclipse.

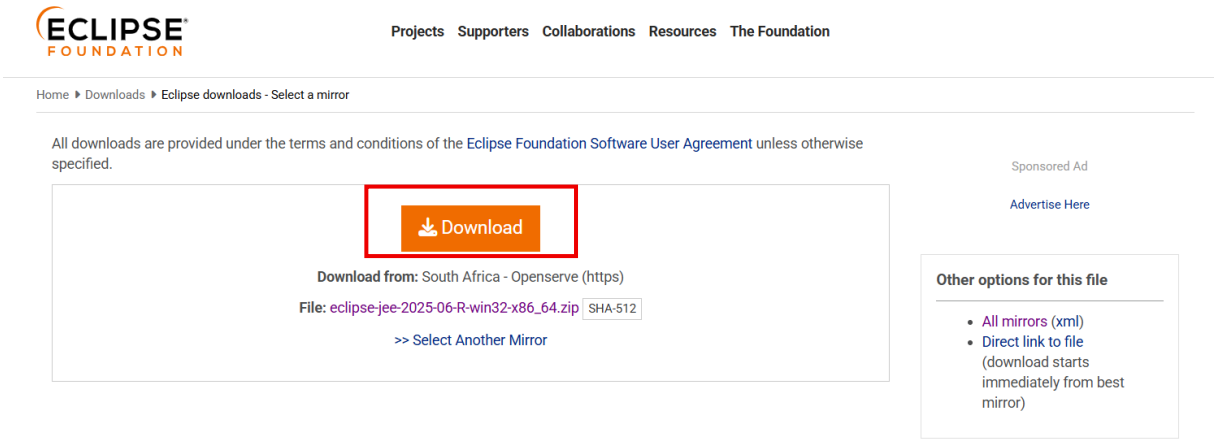
Remarque : L'instance SAP BTP utilisée provient d'un **accès fourni dans le cadre d'une autre formation SAP**, ce qui m'a permis de suivre l'intégralité du parcours RAP dans un environnement cloud conforme.

1.2 Étapes de configuration dans Eclipse

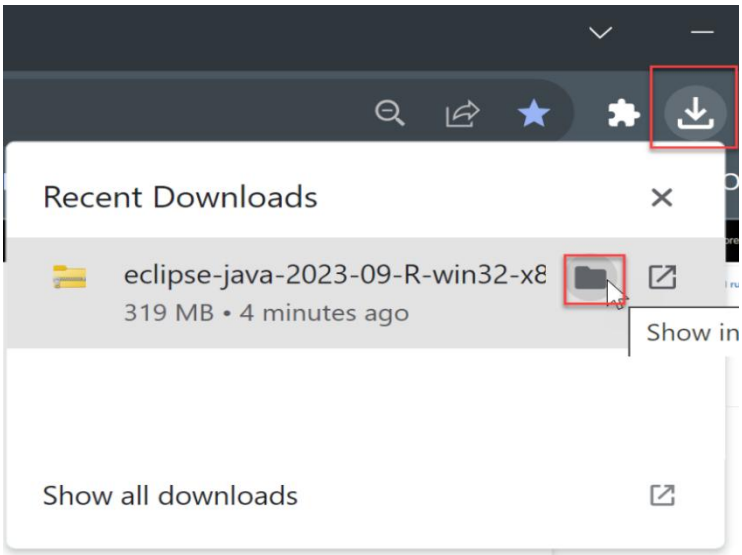
1.2.1 Installation d'Eclipse et des outils ABAP

Téléchargez **Eclipse IDE for Java Developers** depuis https://www.eclipse.org/downloads/packages/ .

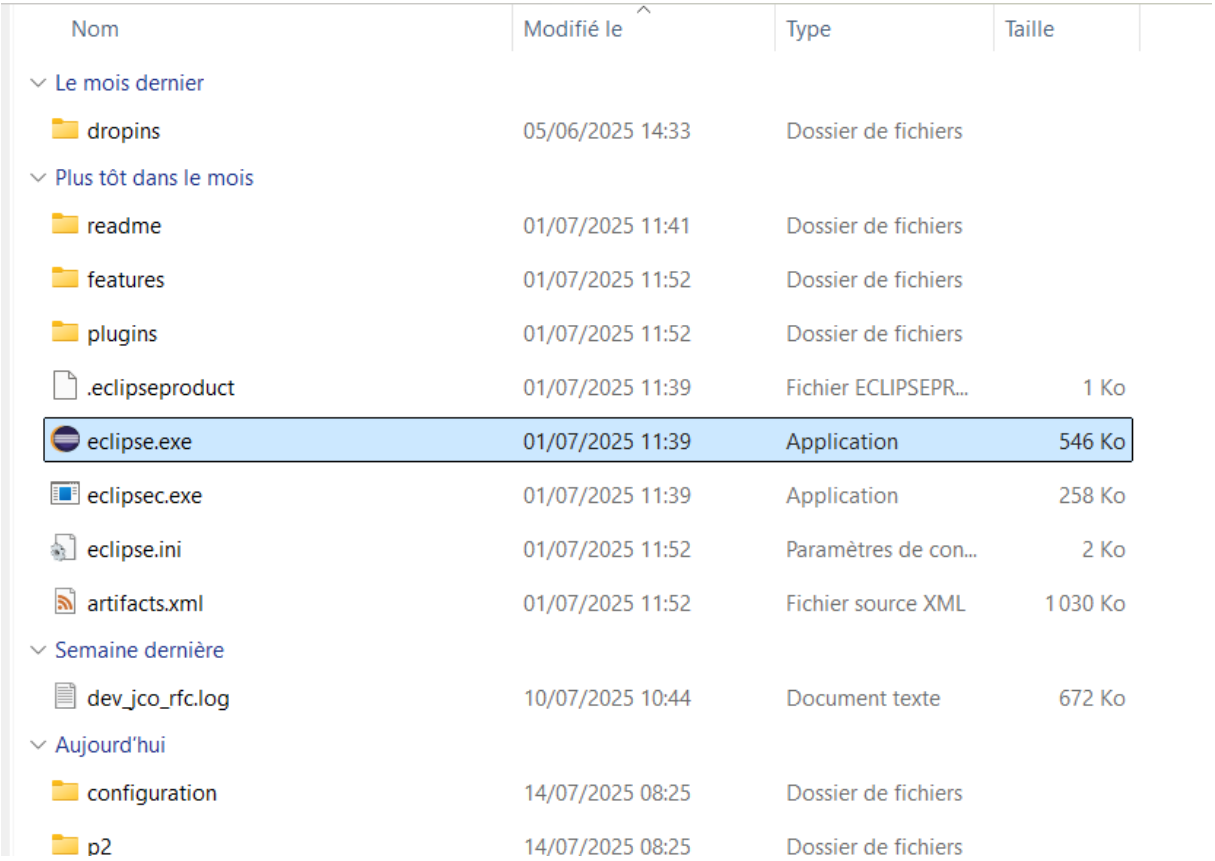




Cliquez sur l'icône de téléchargement de votre navigateur, sélectionnez le symbole du dossier et extrayez le fichier Eclipse.



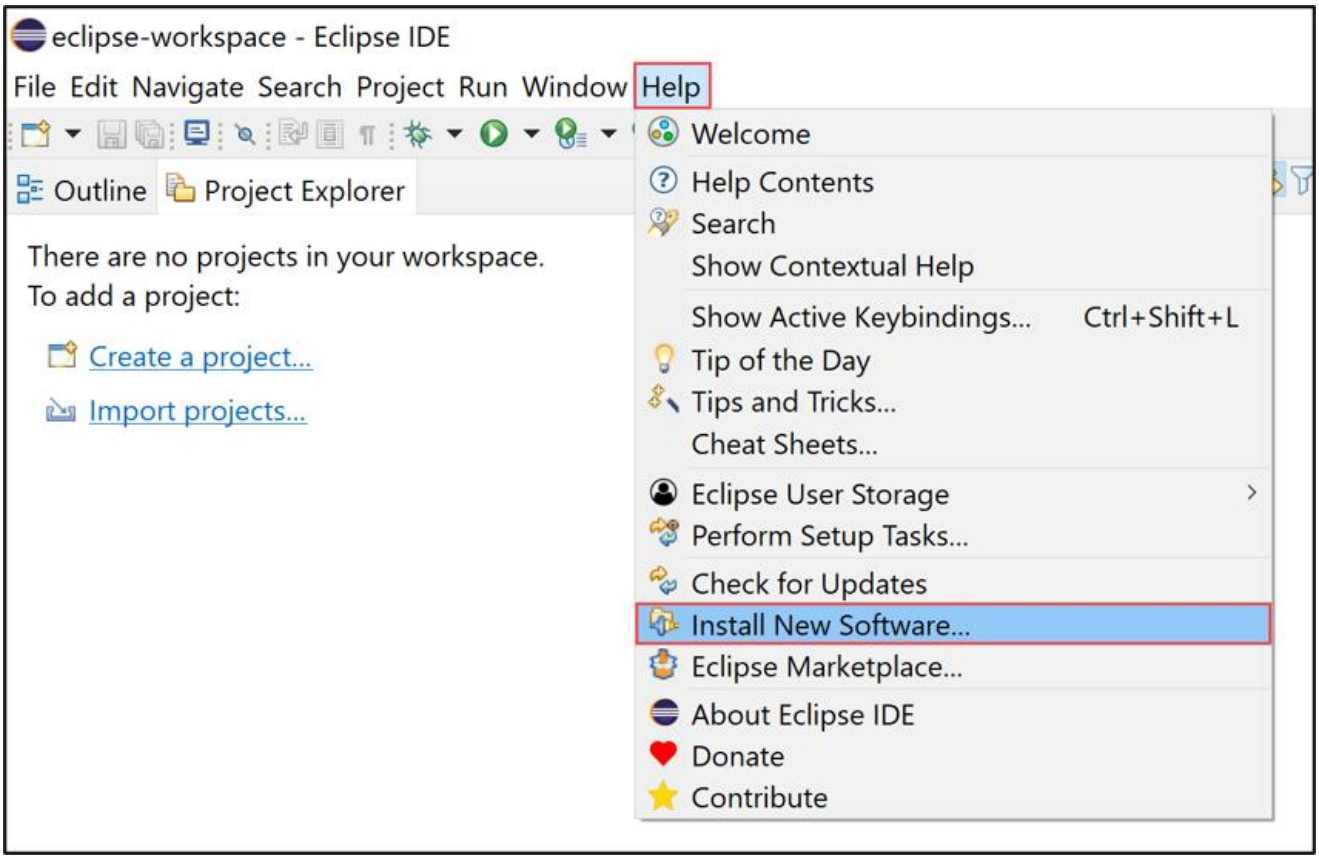
Ouvrez le dossier **Eclipse**, double-cliquez sur **eclipse.exe** pour exécuter l'application et lancer votre espace de travail



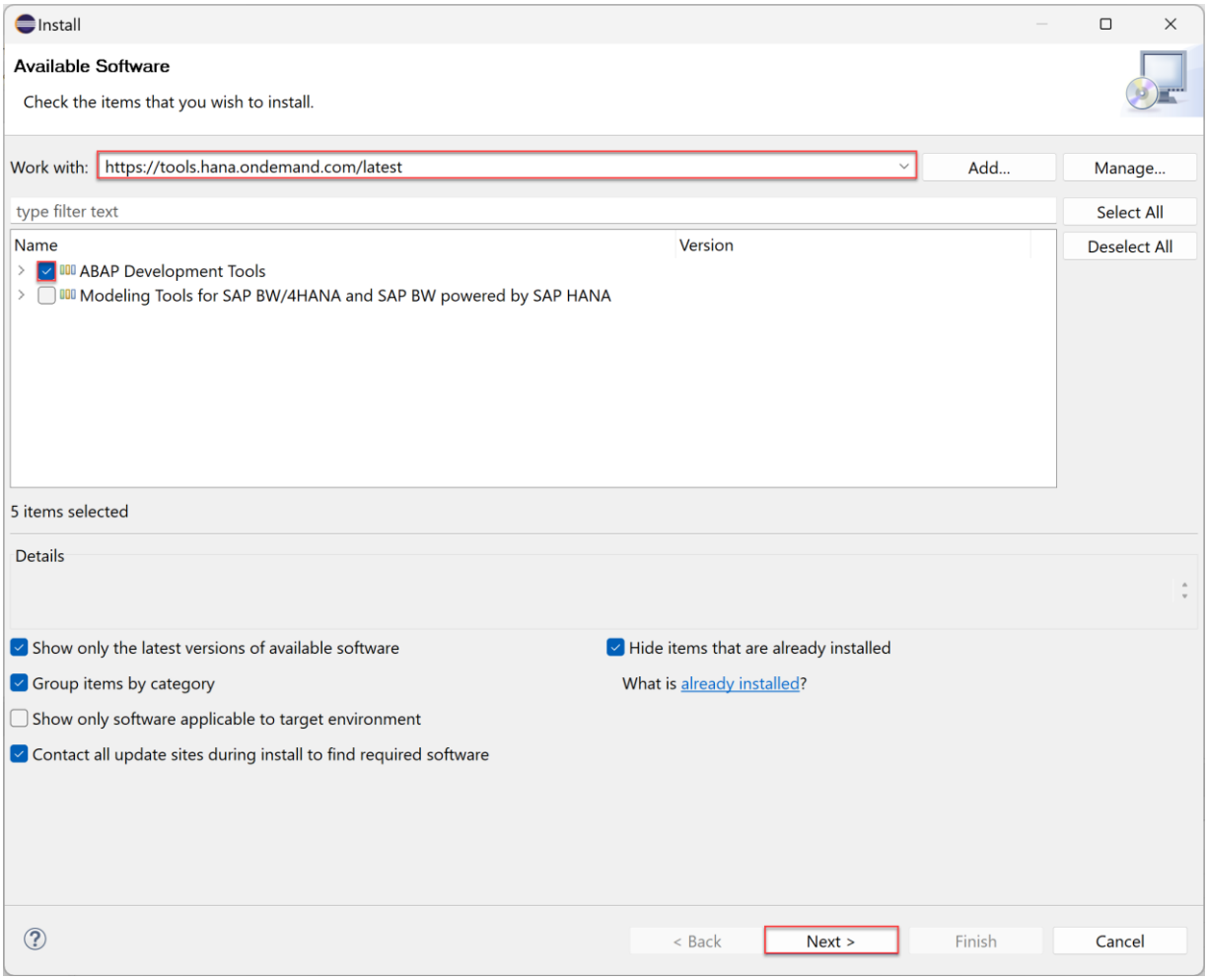
Nom	Modifié le	Type	Taille
Le mois dernier			
dropins	05/06/2025 14:33	Dossier de fichiers	
Plus tôt dans le mois			
readme	01/07/2025 11:41	Dossier de fichiers	
features	01/07/2025 11:52	Dossier de fichiers	
plugins	01/07/2025 11:52	Dossier de fichiers	
.eclipseproduct	01/07/2025 11:39	Fichier ECLIPSEPR...	1 Ko
eclipse.exe	01/07/2025 11:39	Application	546 Ko
eclipsec.exe	01/07/2025 11:39	Application	258 Ko
eclipse.ini	01/07/2025 11:52	Paramètres de con...	2 Ko
artifacts.xml	01/07/2025 11:52	Fichier source XML	1 030 Ko
Semaine dernière			
dev_jco_rfc.log	10/07/2025 10:44	Document texte	672 Ko
Aujourd'hui			
configuration	14/07/2025 08:25	Dossier de fichiers	
p2	14/07/2025 08:25	Dossier de fichiers	

1.2.2 Installation du plugin ABAP Development Tools (ADT) :

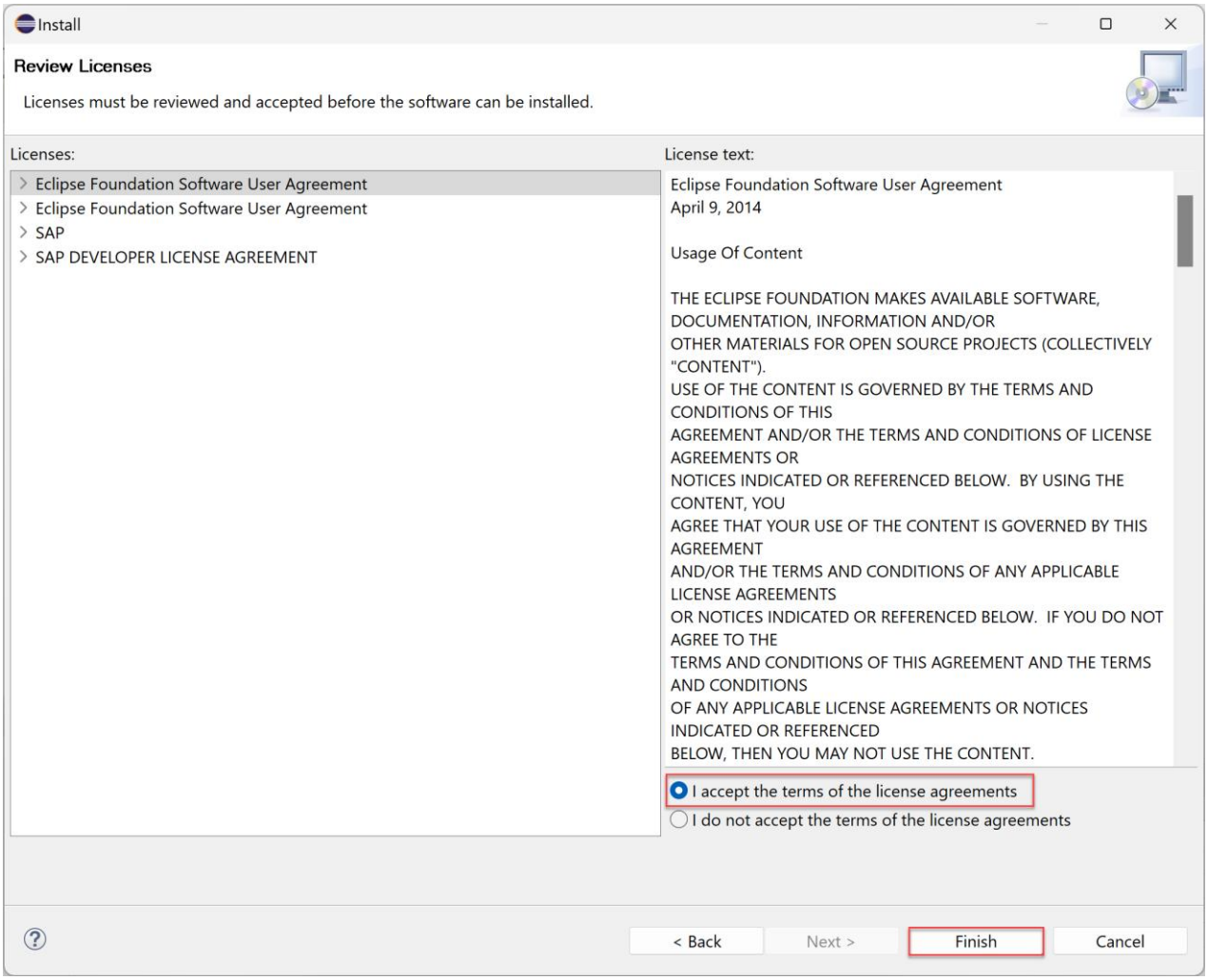
Sélectionner **Help** > **Install New Software**


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Outline
Project Explorer
There are no projects in your workspace.
To add a project:
Create a project...
Import projects...
Welcome
Help Contents
Search
Show Contextual Help
Show Active Keybindings... Ctrl+Shift+L
Tip of the Day
Tips and Tricks...
Cheat Sheets...
Eclipse User Storage
Perform Setup Tasks...
Check for Updates
Install New Software...
Eclipse Marketplace...
About Eclipse IDE
Donate
Contribute

URL du dépôt : https://tools.hana.ondemand.com/latest



Acceptez **le contrat de licence** et cliquez sur **Terminer**.



1.2.3 Configuration d'ADT pour l'utilisation avec SAP Build (Link Handler)

Lorsque l'on souhaite démarrer Eclipse ADT directement depuis SAP Build via un raccourci (eclipse+command), il est nécessaire que cette fonctionnalité soit correctement configurée dans Eclipse. Cette technique repose sur un **"Link Handler"**, qui permet à un outil externe (comme SAP Build) d'ouvrir la bonne instance d'Eclipse avec ADT.

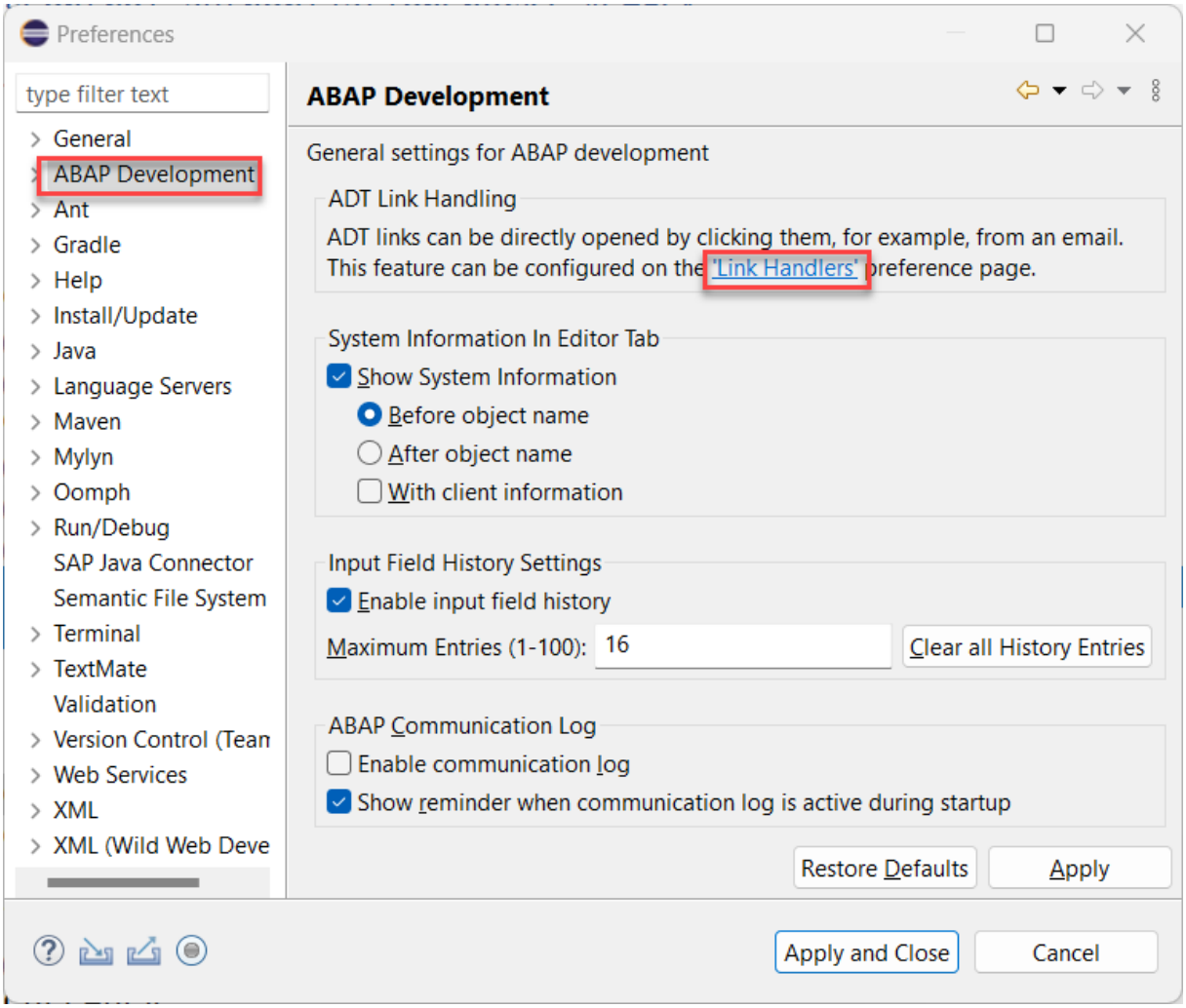
Pourquoi c'est important ?

- Si plusieurs versions d'Eclipse sont installées, SAP Build doit savoir laquelle contient ADT.

- Le lien eclipse+command://... permet d'ouvrir directement un objet (CDS, service, etc.) dans Eclipse depuis SAP Build ou SAP Fiori Tools.

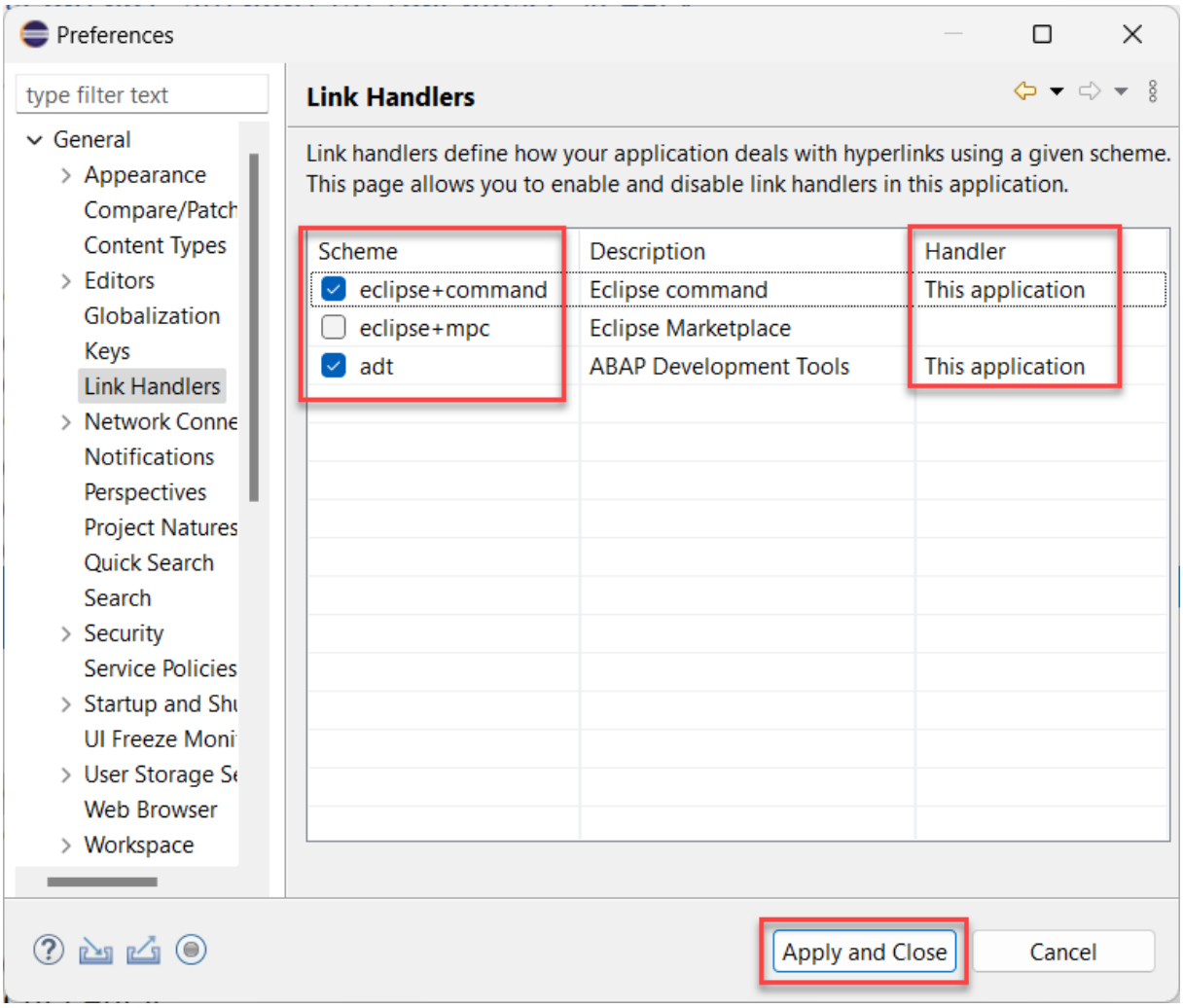
Étapes de configuration du Link Handler (sous Windows)

1. Ouvrez Eclipse ADT (installé lors des étapes précédentes).
2. Dans le menu Eclipse, allez dans :
   Window > **Preferences**
3. Dans la colonne de gauche, développez :
   ABAP Development > **Link Handlers**



4. Vous verrez deux options dans la liste : **Adt** et **eclipse+command**

5. **Cochez les deux cases** pour indiquer que cette instance d'Eclipse doit être utilisée pour gérer ces liens :



Résultat attendu

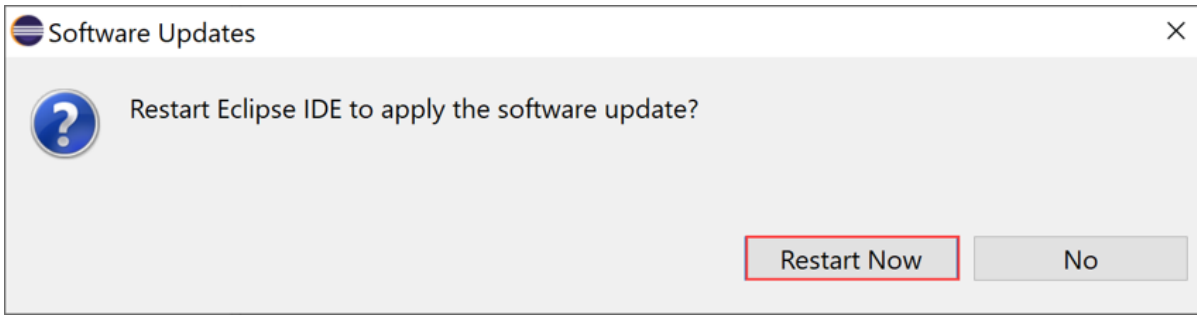
En cochant ces options, votre système saura automatiquement utiliser la bonne instance Eclipse (avec ADT) lorsqu'un lien SAP Build ou Fiori Tools est cliqué.

Remarques :

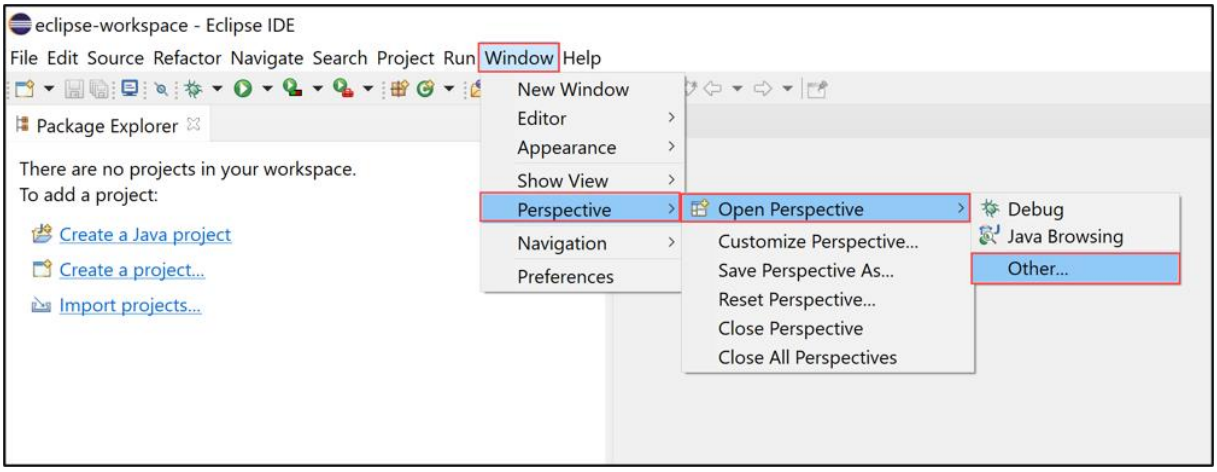
- Sur macOS, cette configuration est faite automatiquement lors de l'installation d'ADT.
- Sur Windows, cette étape est indispensable surtout si vous avez plusieurs versions d'Eclipse installées.

Cliquez sur **Select All and Trust Selected.**

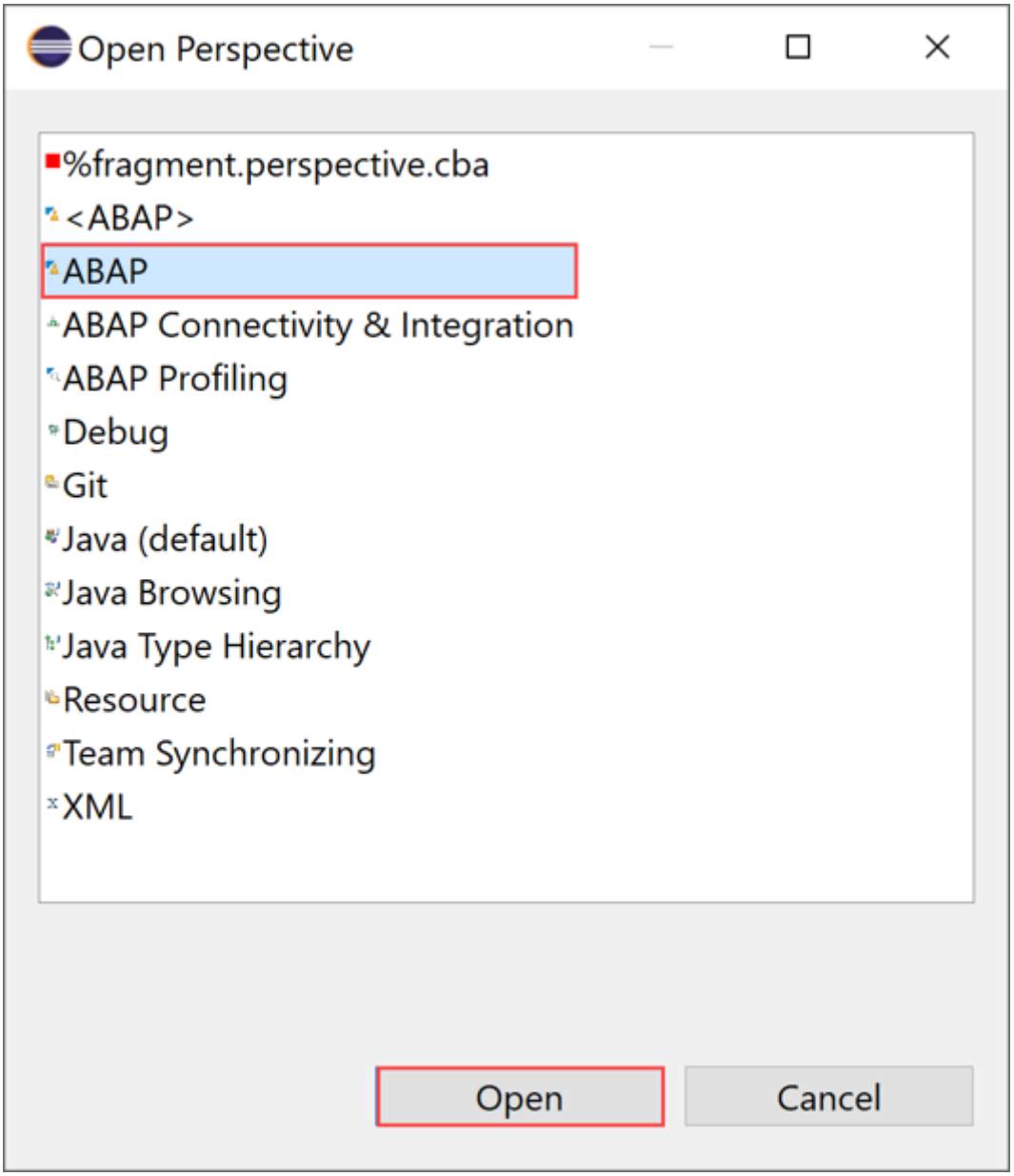
Cliquez sur **Restart Now**.



Maintenant ADT est installé. Passez à ABAP perspective. Ensuite sélectionnez **Window** > **Perspective** > **Open Perspective** > **Other**.



Puis sélectionnez **ABAP** et cliquez sur **Open**.



1.3 Création d'un projet ABAP Cloud via SAP Build

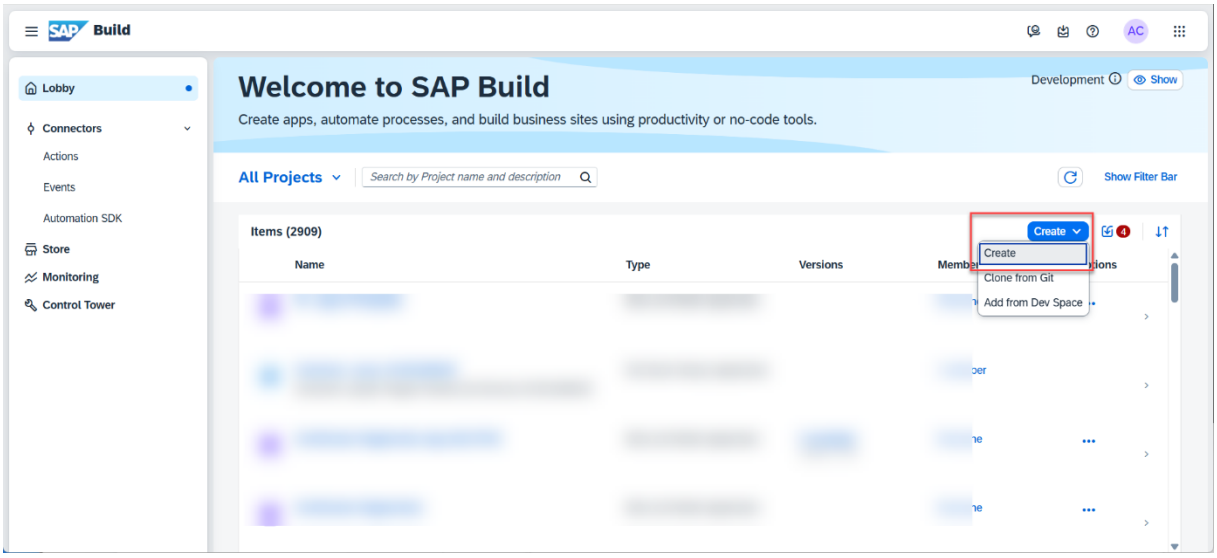
Maintenant que ADT (ABAP Development Tools) est installé et configuré avec le **link handler**, Vous pouvez commencer le développement dans SAP Build. Cette étape permet de **créer un projet Full-Stack ABAP** dans la **SAP Build Lobby**, ainsi qu'un **package dans Eclipse/ADT**, lié à votre environnement ABAP Cloud sur SAP BTP.

Étapes de création du projet dans SAP Build

1. **Ouvrez la SAP Build Lobby**.

   https://sap-build-eu10-trial-2.eu10.build.cloud.sap/lobby

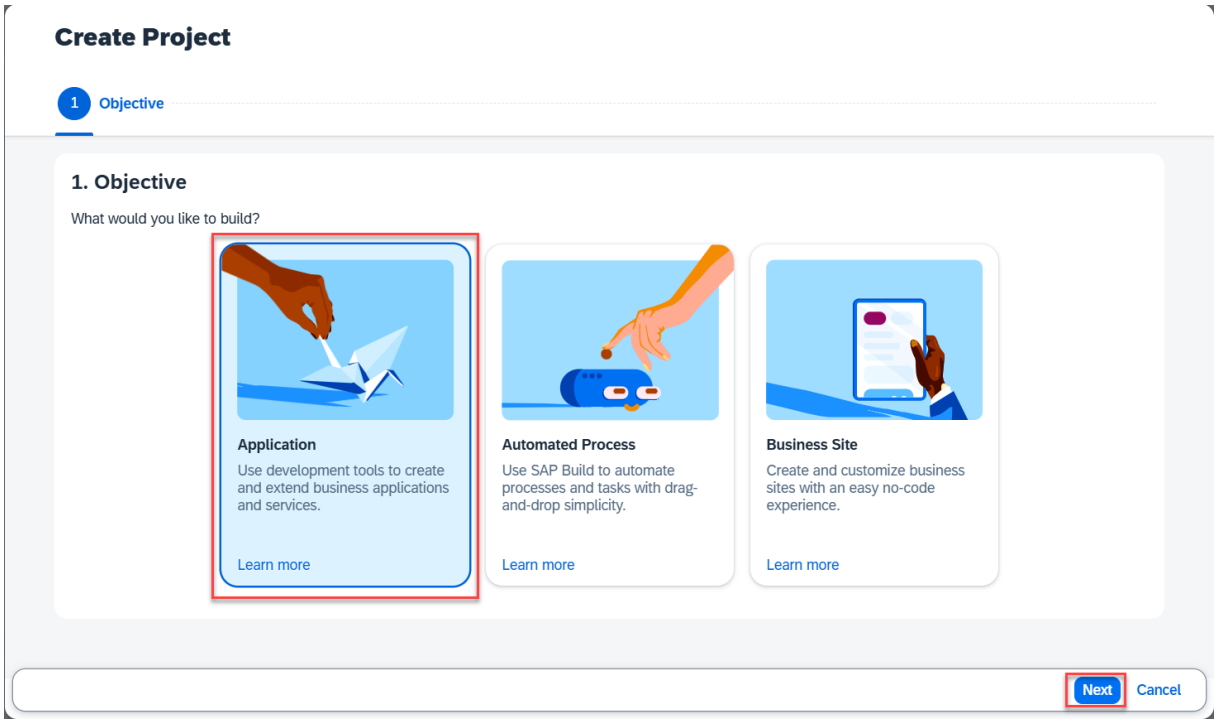
   Une fois connecté, cliquez sur le bouton **"Create"** et choisissez à nouveau **"Create"**)dans le menu déroulant.



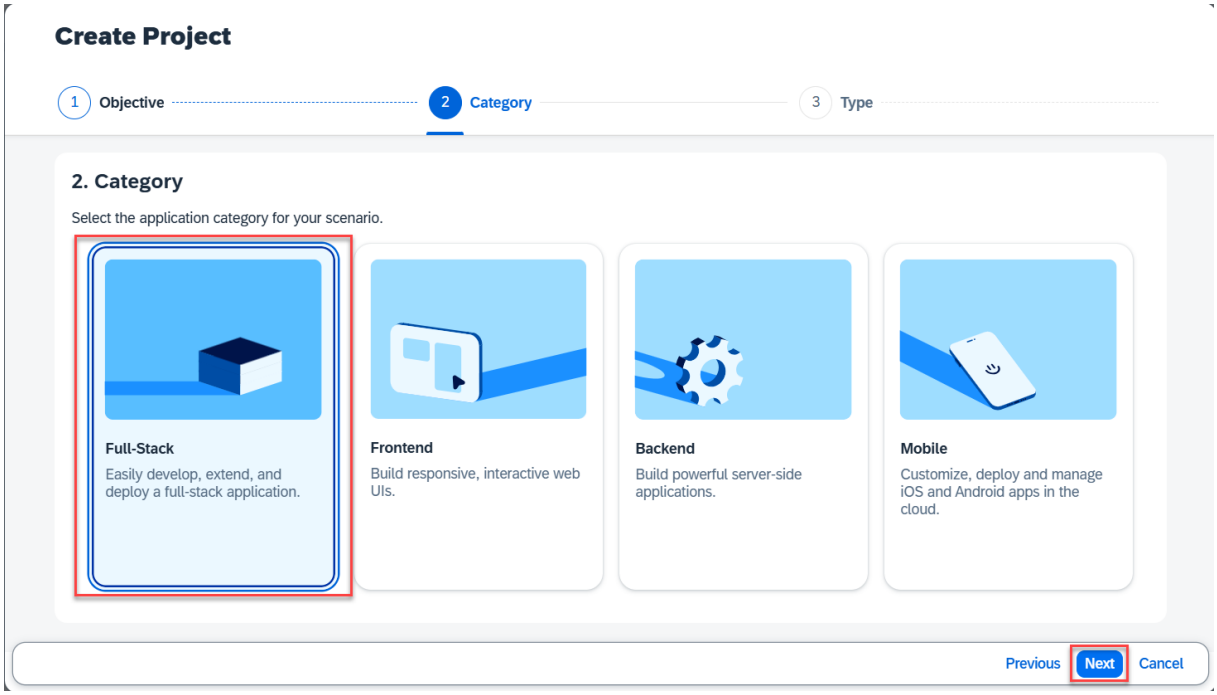
---

2. **Sélection de la catégorie de projet**

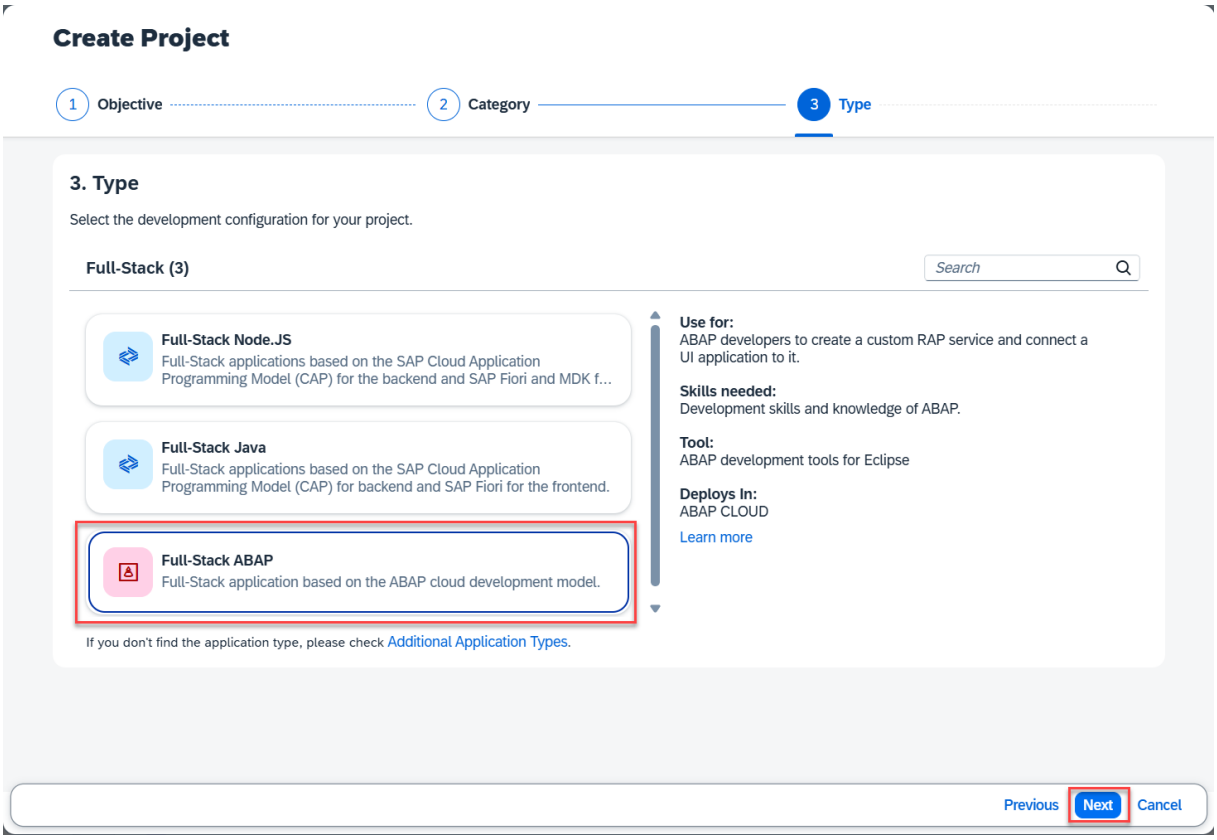
- Choisissez le **tile "Application"**, puis cliquez sur **Next**.



- Choisissez la catégorie : **Full-Stack** à gauche, puis cliquez sur **Next**.

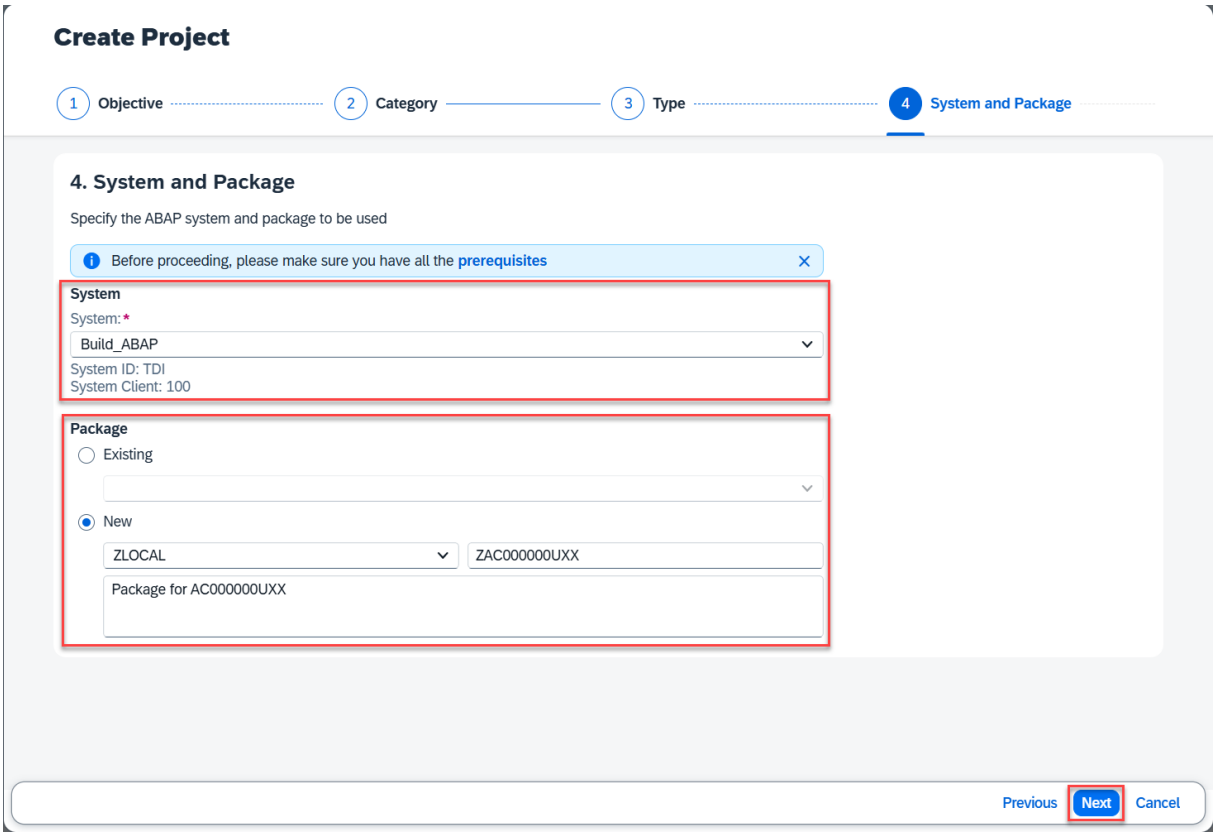


- Type d'application : sélectionnez **Full-Stack ABAP** (3e entrée), puis cliquez sur **Next**.



Connexion au système ABAP Cloud

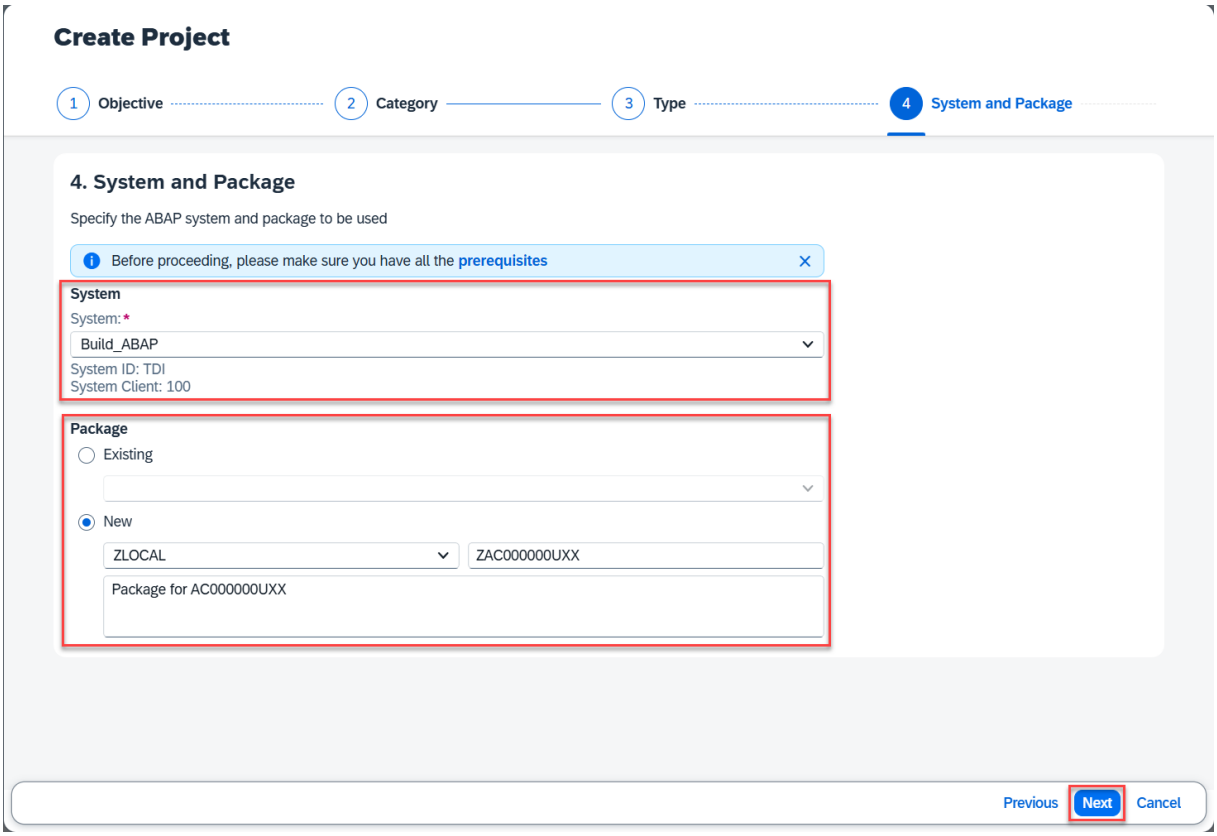
- Le système SAP BTP ABAP sera détecté automatiquement.
- Vous verrez un système nommé comme Build_ABAP, avec un **System ID** (ex. TDI) et un **client** (ex. 100).
- Sélectionnez ce système et continuez.



Création du Package ABAP

Vous allez maintenant créer un **package technique dans Eclipse**, directement via SAP Build :

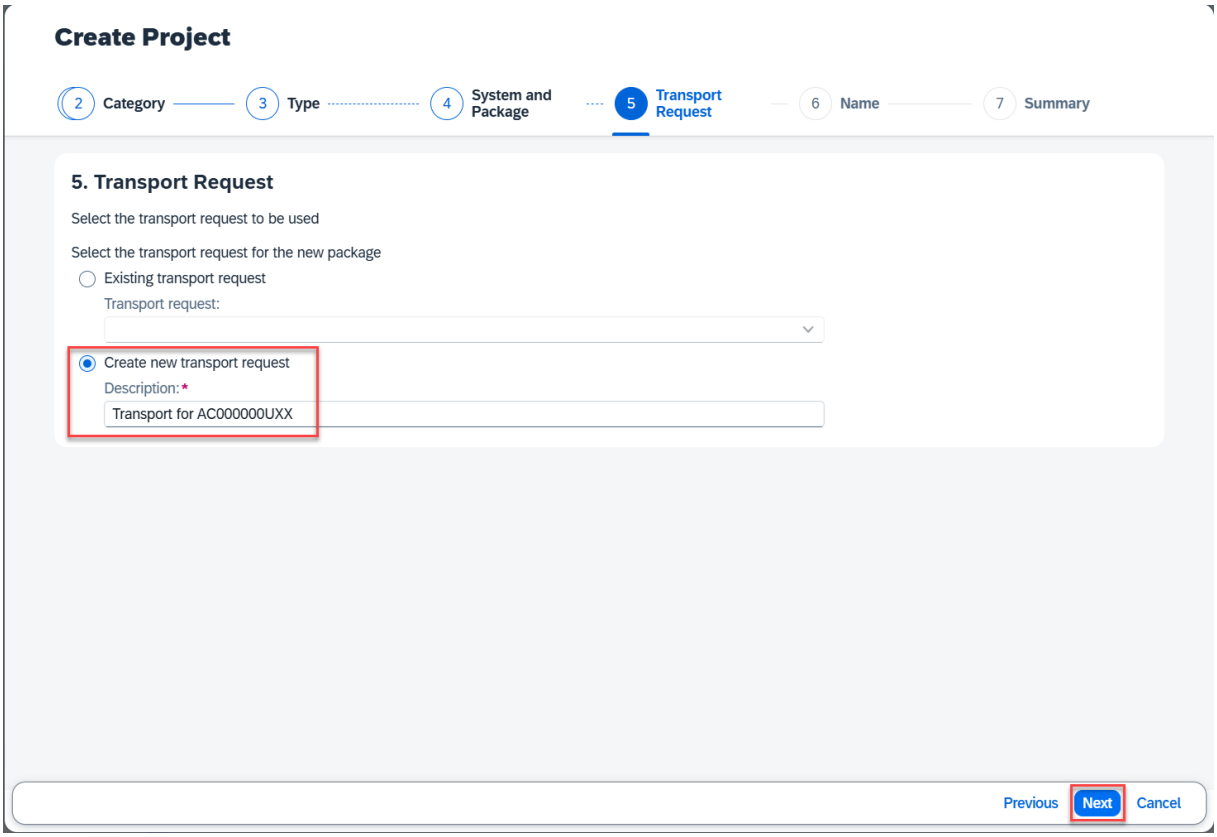
Champ	**Valeur**
Superpackage	ZLOCAL
Nom du package	ZGE200590
Description	Package for ZGE200590



Cliquez sur **Next** une fois terminé.

Création d'une requête de transport

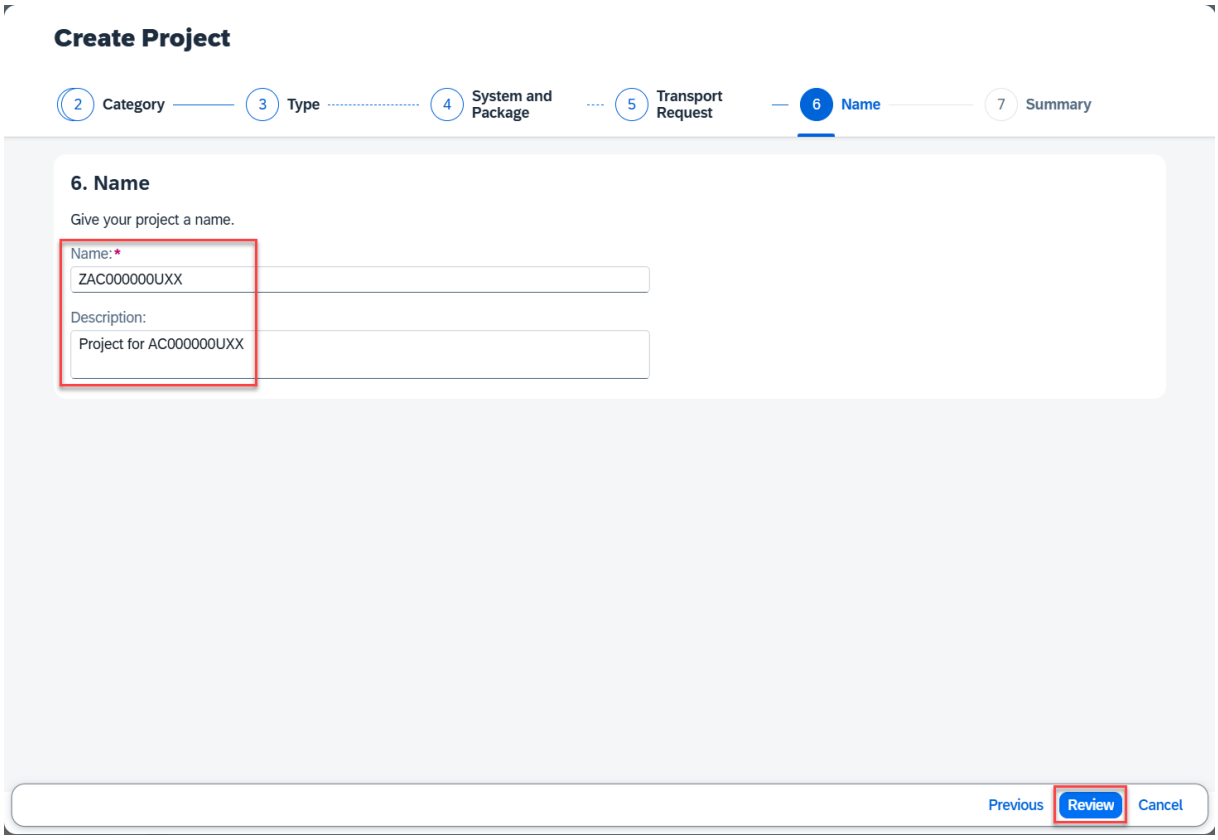
- Sélectionnez **Create a new transport request**
- Description : Transport for GE200590



Cliquez sur **Next**.

Nommer le projet ABAP

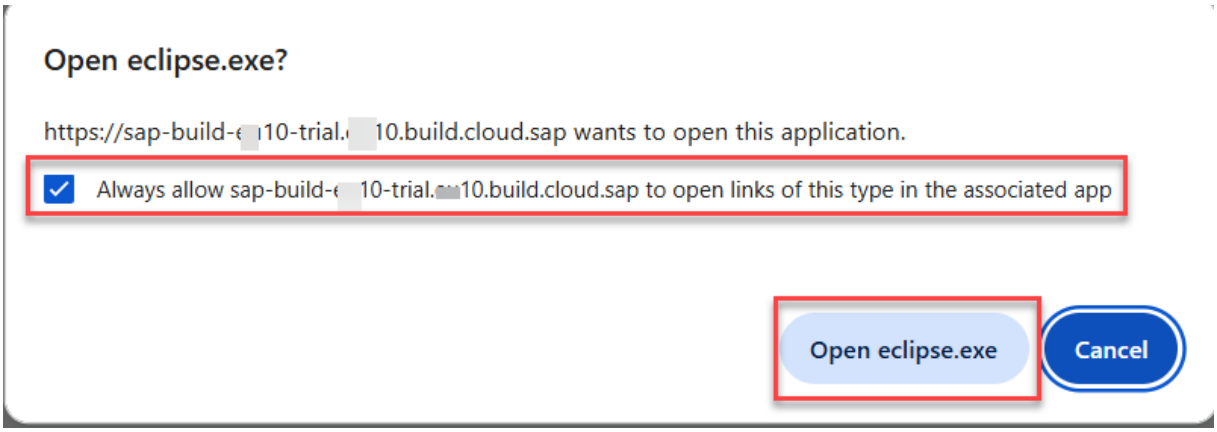
Champ	Valeur
Project Name	ZGE200590
Description	Project for GE200590

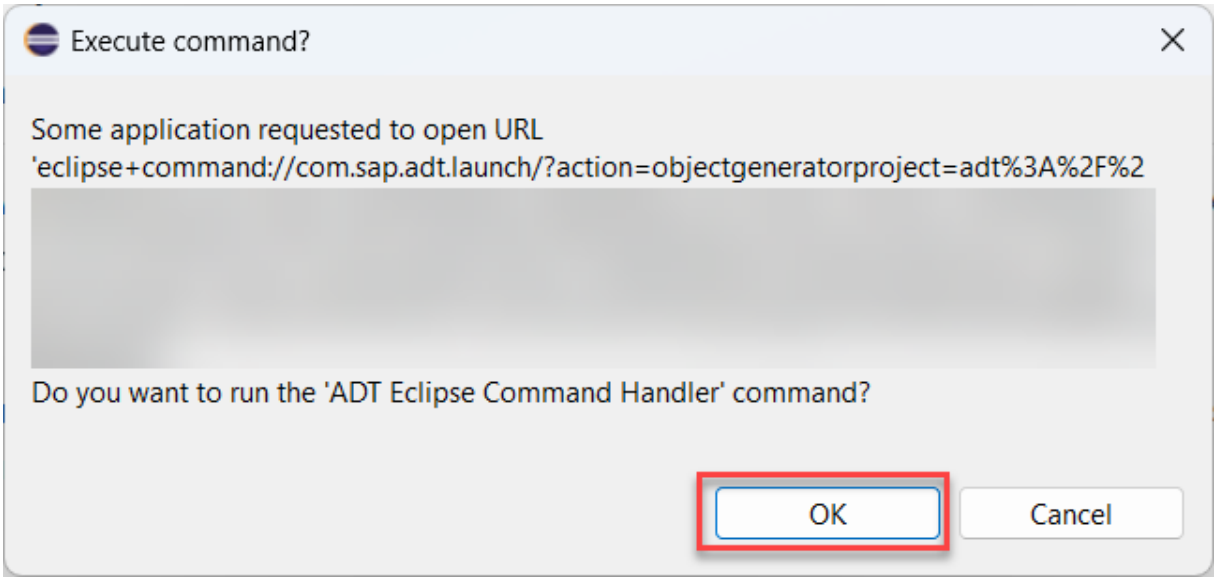


Cliquez sur **Review**, vérifiez les données, puis cliquez sur **Create**.

Lancement d'Eclipse ADT depuis SAP Build

1. Une fenêtre apparaîtra te demandant si vous autorisez SAP Build à ouvrir Eclipse.
2. Cochez **"Always allow links to be opened by this app"**.
3. Cliquez sur **"Open eclipse.exe"**.



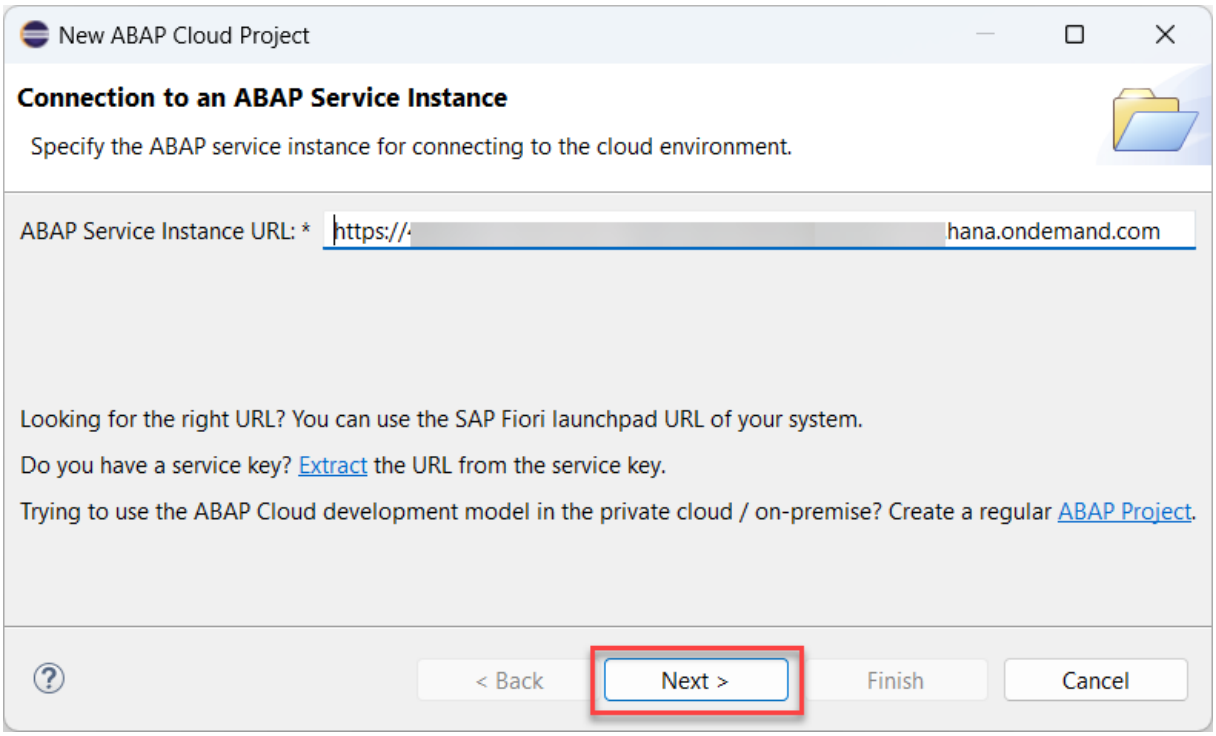


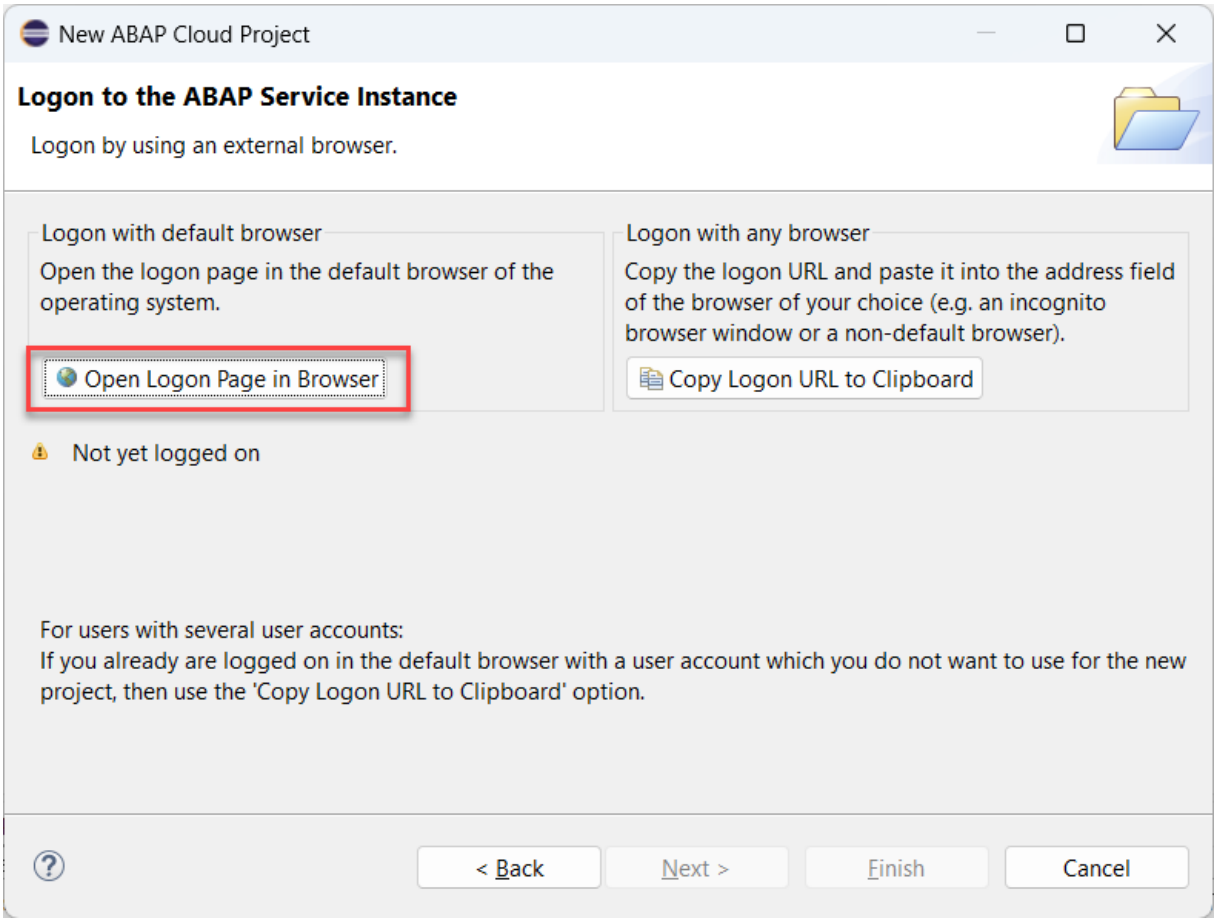
Grâce au **link handler**, votre Eclipse avec ADT se lance automatiquement et ouvre l'assistant de création de projet ABAP Cloud.

Finaliser la création du projet dans Eclipse

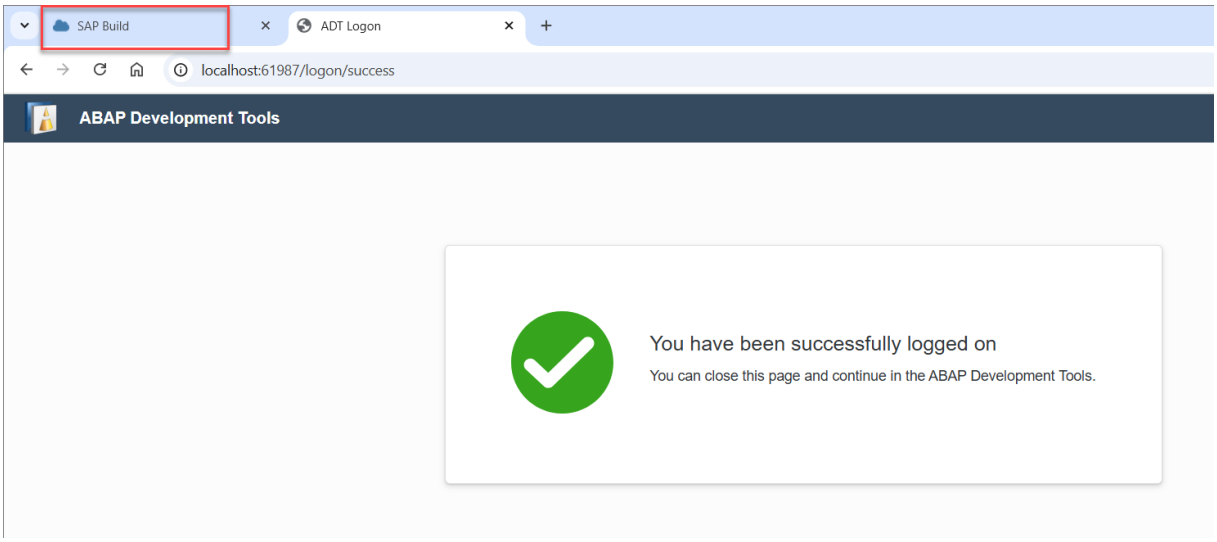
1. L'assistant **"New ABAP Cloud Project"** s'ouvre.

2. Cliquez sur **Next**, puis **Open Logon Page in Browser**.

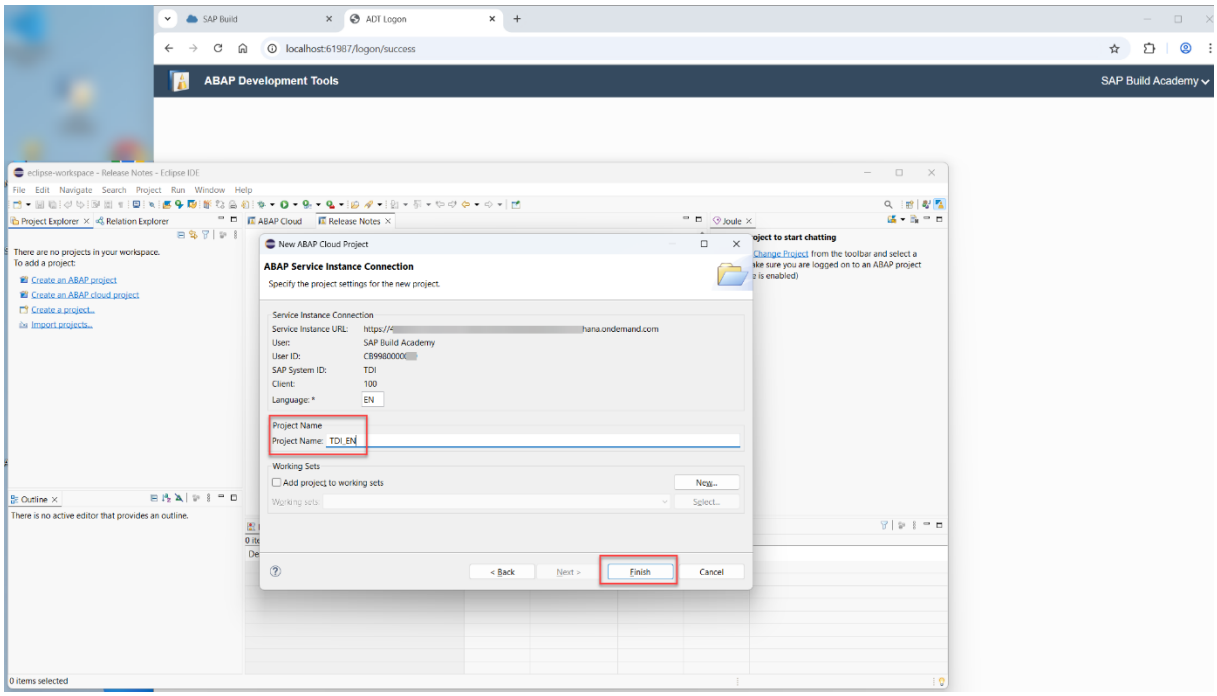




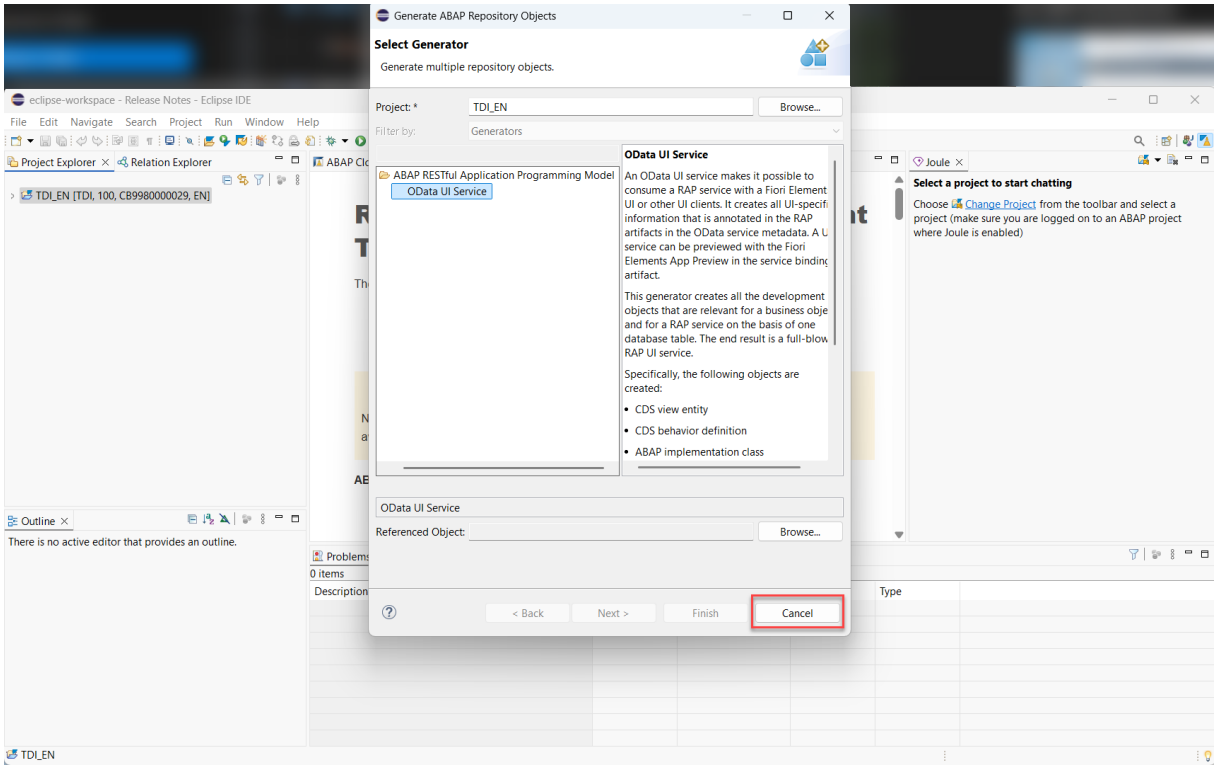
3. Une page SAP s'ouvre, affiche **Logon Success**.



4. Retournez dans Eclipse et cliquez sur **Finish**.

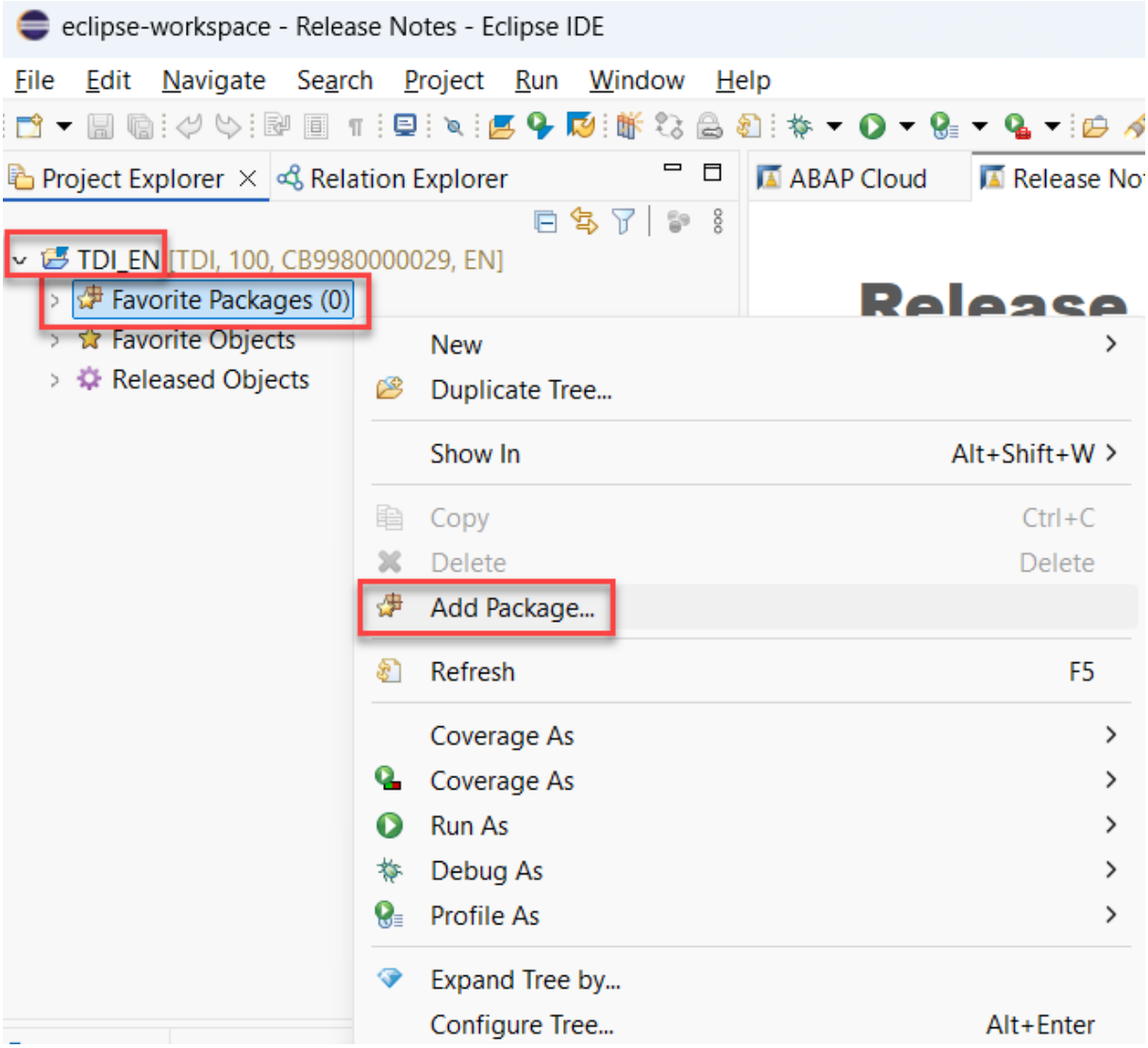


5. **Annulez le wizard UI Service** proposé à la fin (nous ne l'utilisons pas ici).



Ajouter votre package à vos favoris

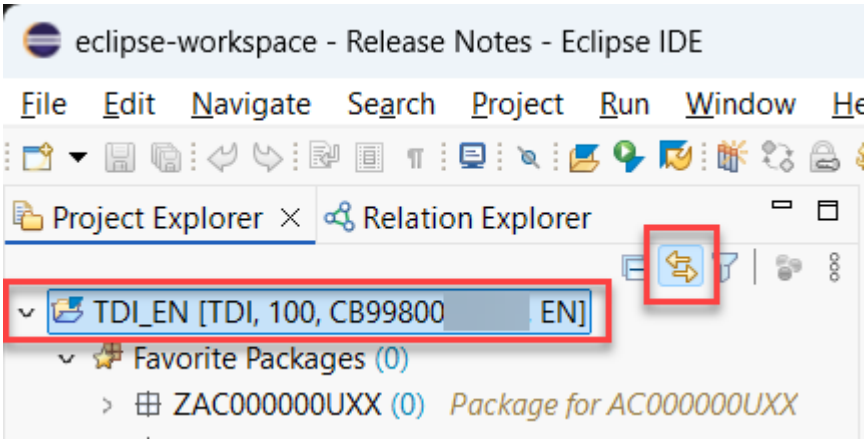
1. Dans l'onglet **Project Explorer**, faites clic droit sur **Favorite Packages > Add Package…**
2. Sélectionnes ZGE200590 (le package créé plus tôt).
3. Cliquez sur **OK**.



Lier l'éditeur au navigateur de projet

- Dans le **Project Explorer**, cliquez sur l'icône **"Link with Editor"** (en haut).
- Elle doit être surlignée pour activer la synchronisation.

- Cela te permet de voir automatiquement l'objet ouvert dans l'arborescence.



Résultat

Votre projet ABAP Cloud est maintenant :

- Connecté à votre instance SAP BTP,
- Prêt à accueillir vos objets RAP (table, CDS, comportement, etc.),
- Organisé dans le package ZGE200590.

Vous êtes maintenant prêt(e) à commencer le développement de votre premier **Business Object RAP**
.

2. Semaine 1 - Introduction – Modèle de programmation d'applications ABAP RESTful

2.1 Vue d'ensemble (Unité 1)

Objectif de l'unité

Cette première unité a pour but de **poser les bases du modèle RAP** (ABAP RESTful Application Programming Model). Elle fournit une **vue d'ensemble stratégique** de ce qu'est RAP, pourquoi SAP l'a créé, et comment il s'intègre dans la transformation cloud de SAP.

Il n'y a pas d'exercice pratique dans cette unité, mais la compréhension des concepts présentés est essentielle pour la suite de la formation.

2.1.1 Qu'est-ce que RAP ?

Le modèle **RAP (RESTful Application Programming Model)** est une approche moderne proposée par SAP pour :

- Développer des applications **orientées métier** en ABAP ;
- Suivant les **principes REST** (Representational State Transfer) ;
- Destinées à être **exécutées dans le cloud** (SAP BTP – ABAP Environment) ou en local sur S/4HANA.

2.1.2 Pourquoi RAP a été introduit ?

SAP a introduit RAP pour répondre à plusieurs objectifs stratégiques :

Objectif	Détail
Cloud natif	Permet de développer des applications directement dans SAP BTP
Standardisation	Imposer des modèles de développement cohérents et réutilisables
Productivité	Faciliter la génération d'UI et de services sans double effort
Sécurité & gouvernance	Intègre nativement des mécanismes de sécurité, autorisations et gestion des erreurs

---

2.1.3 Composants clés du modèle RAP

RAP est basé sur une **architecture en couches**, reposant sur des objets techniques ABAP modernes :

Couche	Élément RAP	Rôle
Modèle de données	CDS View (Core Data Services)	Représentation des entités métiers (read-only ou projection)
Comportement	Behavior Definition & Implementation	Décrit et implémente la logique métier (création, validation, etc.)
Service	Service Definition & Binding	Expose les entités RAP sous forme d'API OData (REST)
Interface utilisateur	Fiori Elements	UI générée automatiquement à partir du modèle RAP

Résumé de l'unité

- RAP permet de construire des applications cloud modernes en ABAP.
- Il repose sur les principes REST et les technologies ABAP récentes (CDS, Behaviors, OData).
- Cette unité sert à comprendre **le cadre global dans lequel s'inscrit RAP**, avant de plonger dans le développement technique.

2.2 Présentation de l'architecture (Unité 2)

Objectif de l'unité

Cette unité présente l'**architecture globale du modèle RAP**, en expliquant comment les différents éléments (modèle de données, comportement, services, etc.) interagissent entre eux pour construire une application RESTful complète dans un environnement ABAP.

2.2.1 Vue d'ensemble de l'architecture RAP

Cette unité approfondit l'architecture en couches introduite précédemment :

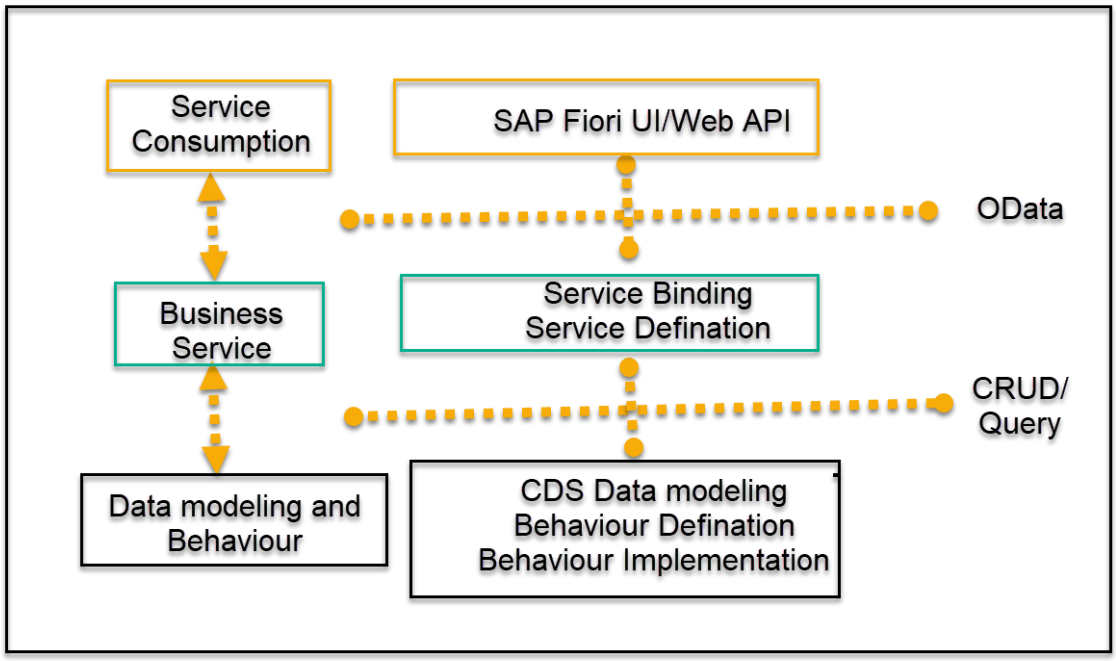


Figure 1 Modèle d'architecture RAP

2.2.2 Description des couches principales

Couche	Élément	Rôle
UI	Fiori Elements / SAP Build	Génération automatique d'interface utilisateur à partir des métadonnées RAP
Service	Service Binding & Definition	Expose les entités RAP sous forme d'API REST (OData V4)
Comportement	Behavior Definition & Implementation	Gère les opérations CRUD, validations, actions et transactions métier
Modèle de données	CDS View (Root, Projection...)	Représente les données et leurs relations, avec enrichissements (annotations, associations...)
Base de données	Tables ABAP	Stocke physiquement les données manipulées par l'application

2.2.3 Cycle de vie des données dans RAP

1. L'utilisateur déclenche une action (ex : créer un enregistrement) via l'UI.
2. Le service REST envoie la requête vers le **BO (Business Object RAP)**.
3. La **Behavior Implementation** exécute la logique correspondante.
4. Les données sont lues ou modifiées via les **CDS views** et les tables.
5. La réponse est renvoyée via OData, avec ou sans retour utilisateur (messages, erreurs, succès...).

2.3 Technologies impliquées (Unité 3)

Objectif de l'unité

Cette unité présente les **technologies clés utilisées dans le modèle RAP**, qui s'intègrent pour offrir une solution moderne, performante et sécurisée pour le développement ABAP dans le cloud.

2.3.1 Technologies utilisées dans RAP

Technologie	Rôle dans RAP
ABAP	Langage principal utilisé pour coder la logique métier
Core Data Services (CDS)	Définir les vues et modèles de données enrichis
ABAP Development Tools (ADT)	Développement dans Eclipse avec plugin ADT
OData V4	Protocole RESTful utilisé pour exposer les services RAP
SAP Fiori Elements	Génération d'interface utilisateur à partir de métadonnées
SAP Business Technology Platform (BTP)	Environnement d'exécution cloud natif pour RAP
Behavior Definitions (BO)	Décrit les opérations autorisées (create, update, delete...) et la logique transactionnelle

Technologie	Rôle dans RAP
Authorization (CDS Access Control, PFCG)	Gestion de la sécurité et des accès aux entités exposées

---

2.3.2 Synergie des composants

Toutes ces technologies ont été pensées pour fonctionner **de manière intégrée** :

- CDS + Behavior = Modèle Métier sémantique.
- Binding + Service Definition = Exposition sécurisée via API.
- ADT + SAP Build = Développement fluide et productif.
- Fiori Elements = UI générée automatiquement sans développement manuel.

2.4 Présentation de SAP Cloud Platform, environnement ABAP (Unité 4)

Objectif de l'unité

Comprendre le rôle du **SAP BTP – ABAP Environment**, qui est la **plateforme cloud** sur laquelle les applications RAP sont développées et exécutées.

2.4.1 Qu'est-ce que SAP BTP – ABAP Environment ?

C'est un **environnement ABAP dans le cloud**, qui permet :

- L'exécution cloud des applications RAP, intégrant CDS, ADT, et bien d'autres technologies vues précédemment. Des applications avec les technologies RAP, CDS, ADT...
- D'exécuter ces applications dans un contexte cloud sécurisé et scalable.
- De s'affranchir des contraintes d'un système on-premise SAP traditionnel.

2.4.2 Fonctionnalités offertes

Fonctionnalité	Détail
Développement Cloud	Utilisation d'ADT dans Eclipse pour se connecter à l'environnement ABAP BTP
Sécurité intégrée	Gestion des utilisateurs, des autorisations, et des journaux d'accès
Exposition OData	Services accessibles via API RESTful (OData V4)
Transport	Gestion des transports dans le cloud (similaire au TMS on-premise)
Conteneurs de développement	Packages, classes, vues CDS, comportements, services...
Extensions	Intégration possible avec des bots, IA, événements, etc.

2.5 Création de votre première application console ABAP Cloud (Unité 5)

Objectif de l'unité

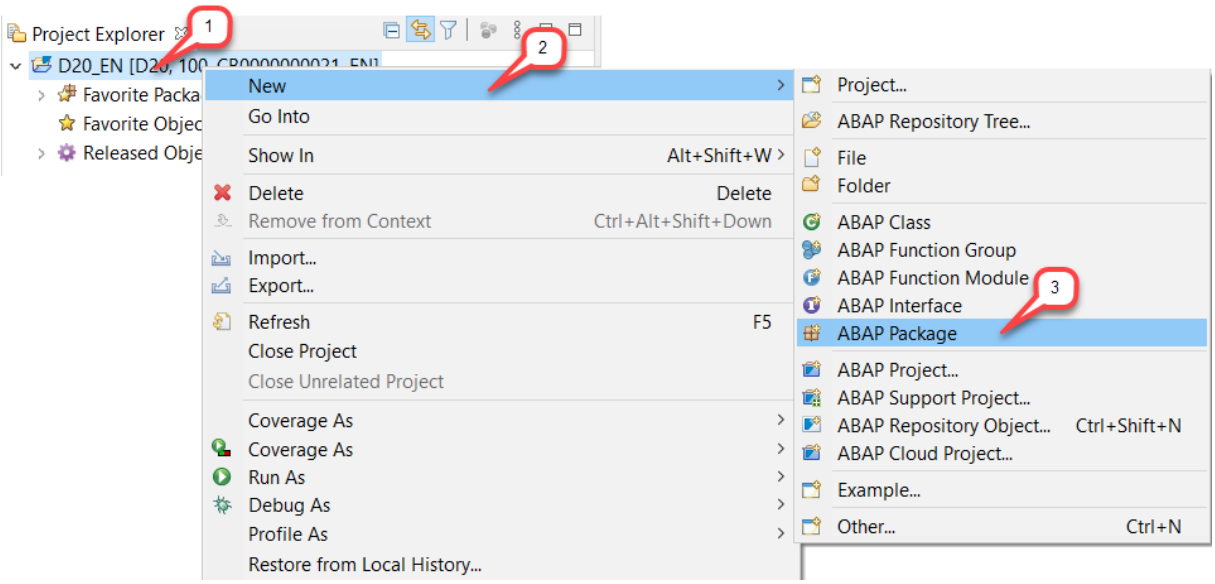
Cette unité pratique marque le début du développement réel dans l'environnement ABAP Cloud. L'objectif est de créer une application console simple, affichant "Hello World!" et le nom de l'utilisateur connecté. Ce projet introduit les bases de l'utilisation d'ADT (ABAP Development Tools) pour développer dans SAP BTP – ABAP Environment.

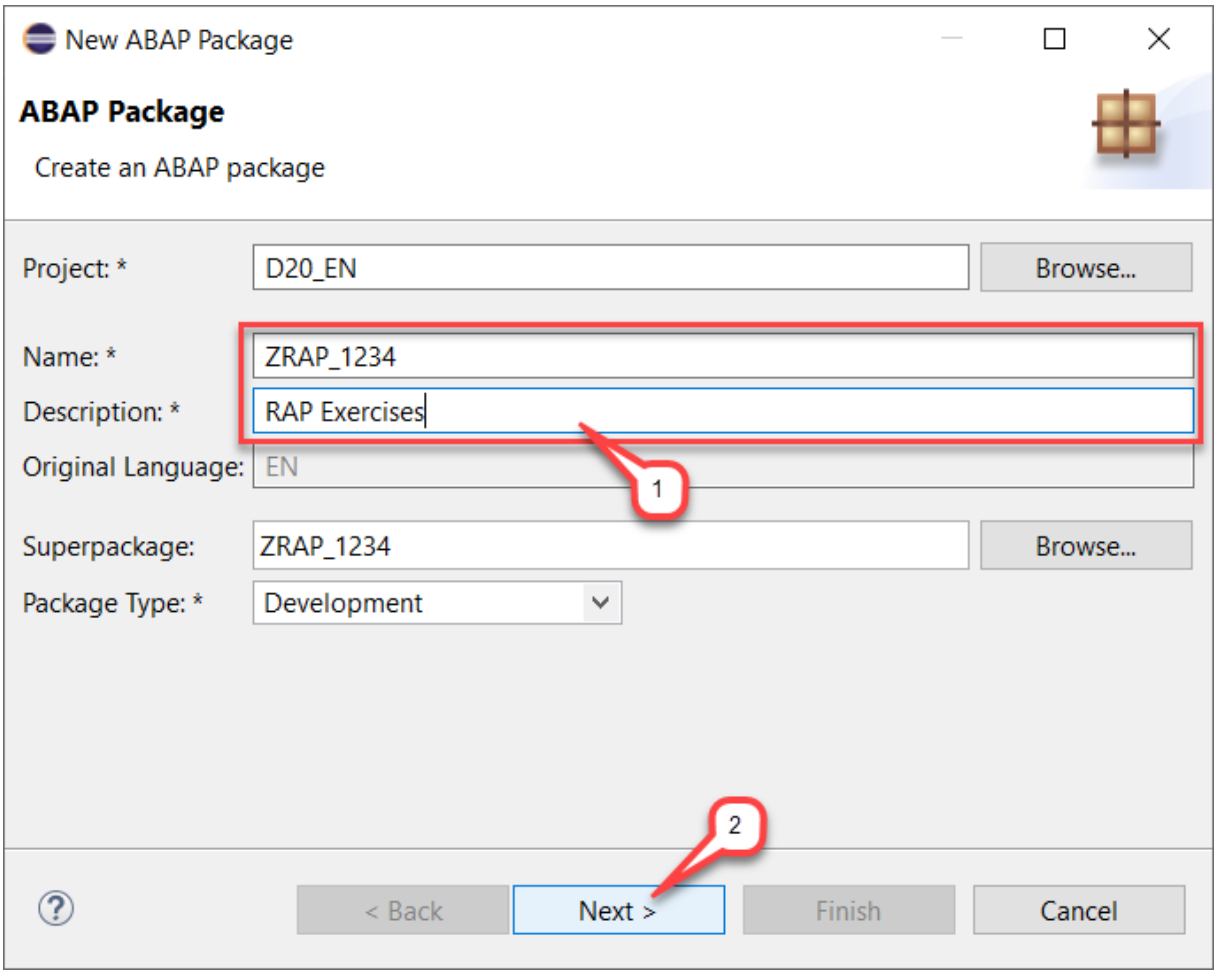
Étapes principales de l'exercice pratique

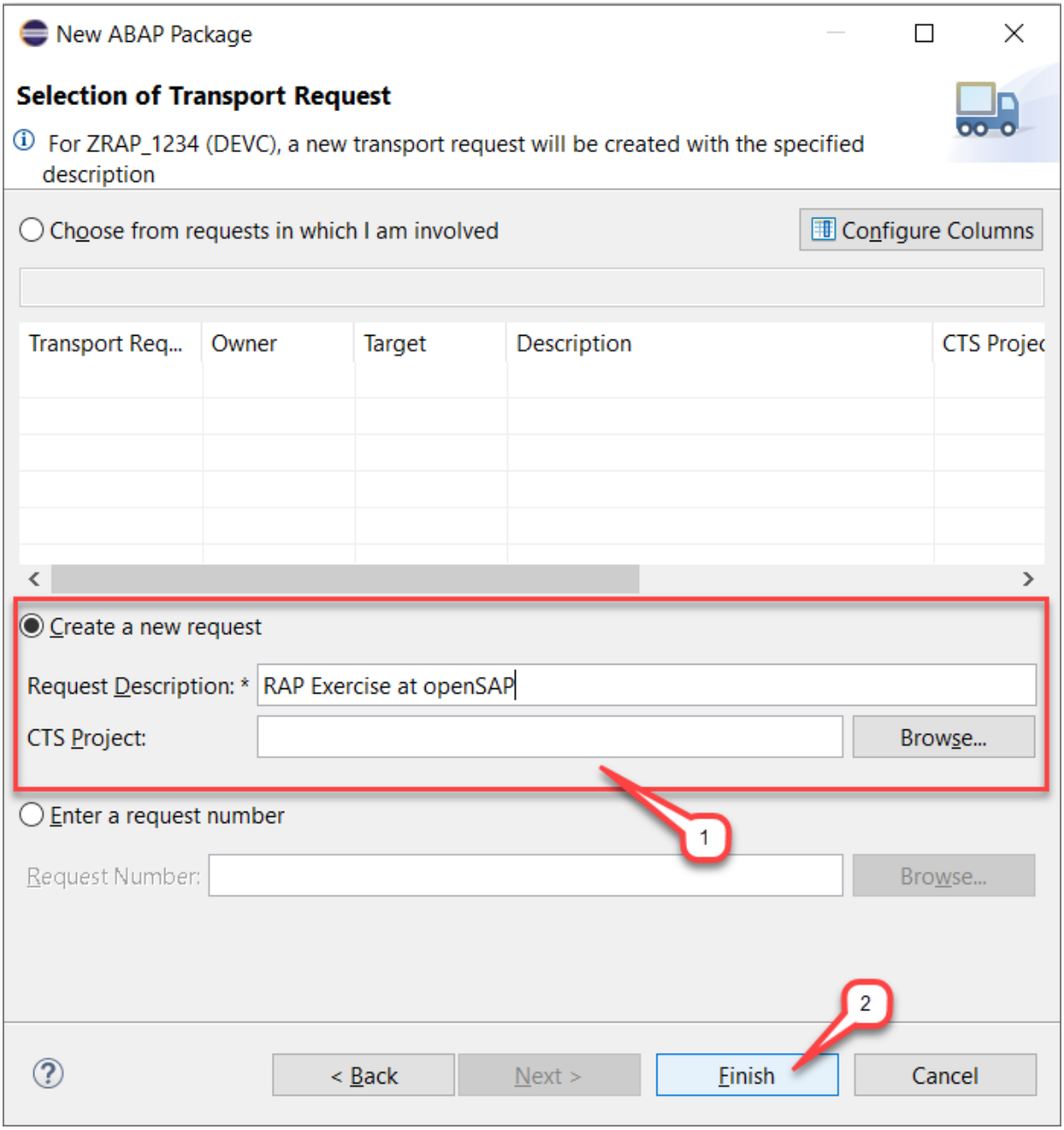
2.5.1 Création d'un package ABAP

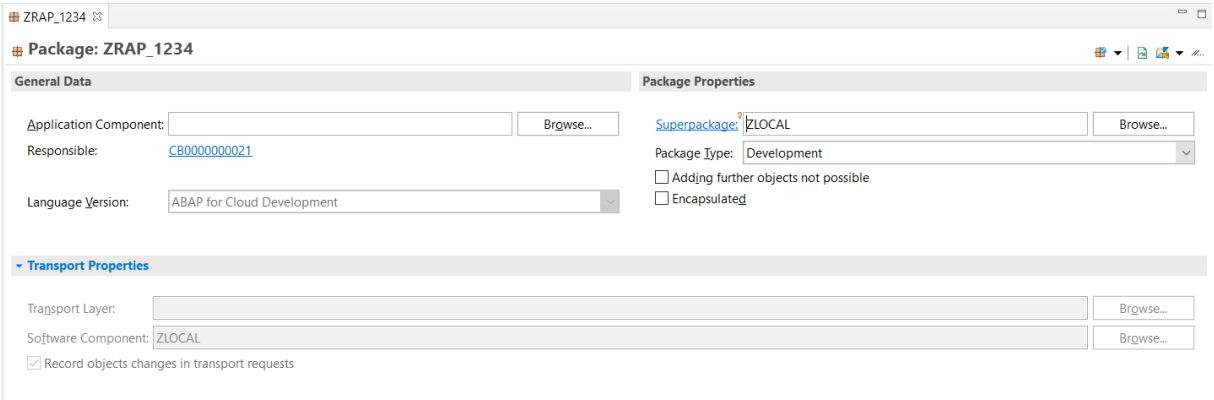
- **Nom du package** : ZRAP_#### (où #### est votre suffixe personnel, ex. 1234)
- **Description** : *RAP Exercises*
- **Superpackage** : ZLOCAL
- **Transport Request** : Créé ou sélectionné depuis SAP Build.

Astuce : Ajoute votre package aux "Favorite Packages" pour un accès rapide.


Project Explorer
D20_EN [D20, 100, CR0000000021, EN]
Favorite Packa
Favorite Objec
Released Obje
1
2
New
Go Into
Show In Alt+Shift+W
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Import...
Export...
Refresh F5
Close Project
Close Unrelated Project
Coverage As
Coverage As
Run As
Debug As
Profile As
Restore from Local History...
Project...
ABAP Repository Tree...
File
Folder
ABAP Class
ABAP Function Group
ABAP Function Module
ABAP Interface
ABAP Package
3
ABAP Project...
ABAP Support Project...
ABAP Repository Object... Ctrl+Shift+N
ABAP Cloud Project...
Example...
Other... Ctrl+N

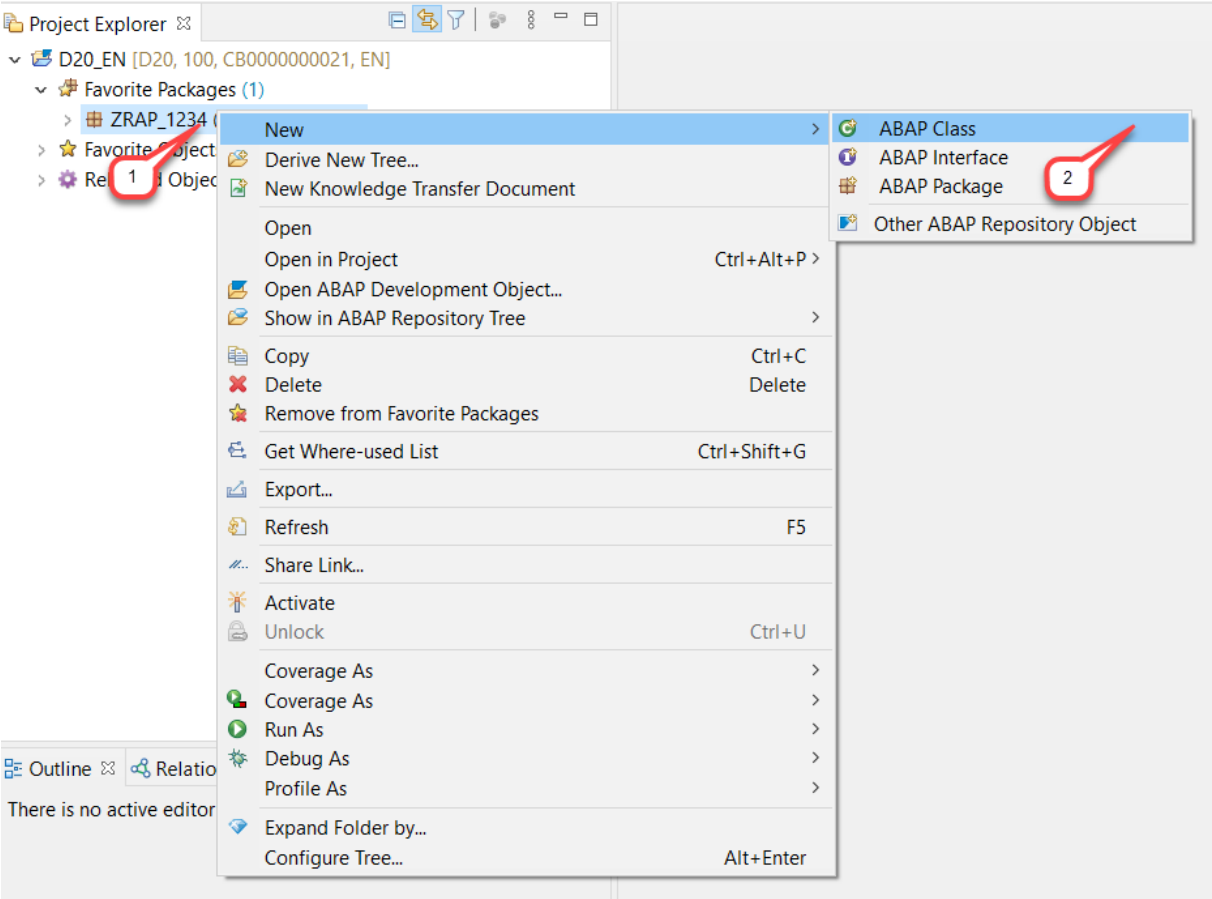

New ABAP Package
ABAP Package
Create an ABAP package
Project: *
D20_EN
Browse...
Name: *
ZRAP_1234
Description: *
RAP Exercises
Original Language:
EN
1
Superpackage:
ZRAP_1234
Browse...
Package Type: *
Development
2
< Back
Next >
Finish
Cancel

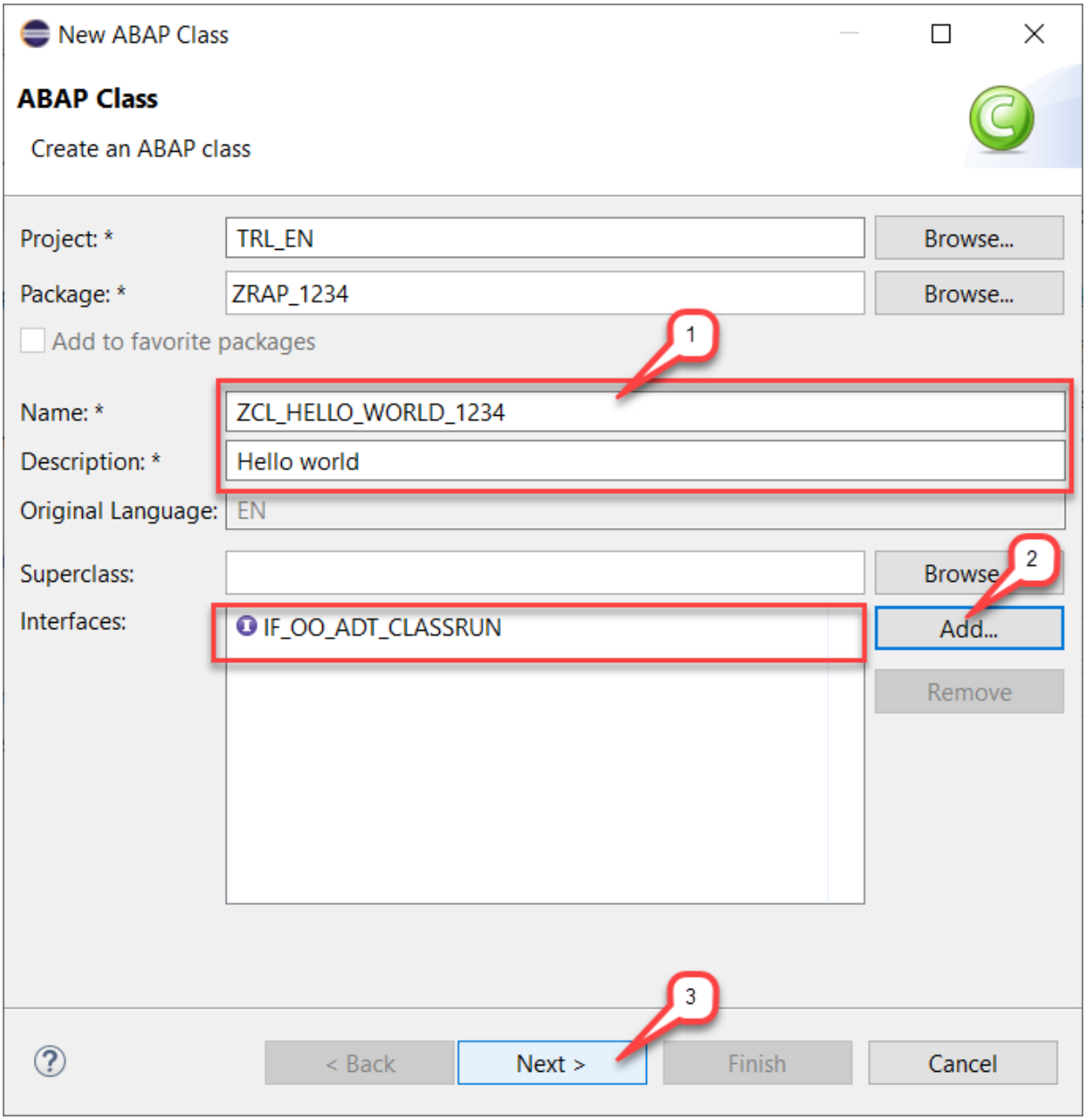

New ABAP Package
Selection of Transport Request
For ZRAP_1234 (DEVC), a new transport request will be created with the specified description
Choose from requests in which I am involved
Configure Columns
Transport Req...
Owner
Target
Description
CTS Projec
Create a new request
Request Description: *
RAP Exercise at openSAP
CTS Project:
Browse...
1
Enter a request number
Request Number:
Browse...
2
< Back
Next >
Finish
Cancel

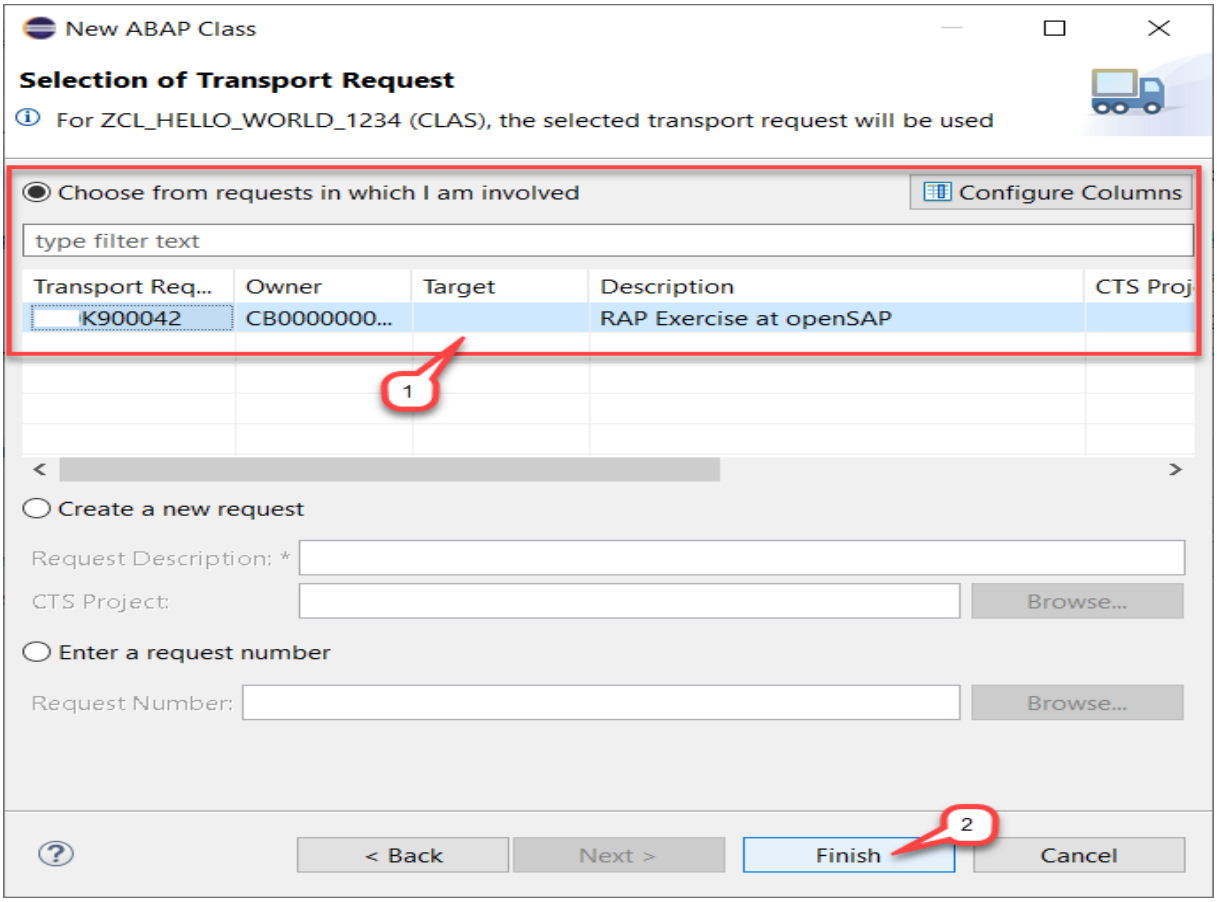


2.5.2 Création d'une classe ABAP Hello World

- **Nom de la classe** : ZCL_HELLO_WORLD_####
- **Description** : *Hello world*
- **Interface implémentée** : IF_OO_ADT_CLASSRUN (permet l'exécution dans la console ABAP)


Project Explorer
D20_EN [D20, 100, CB0000000021, EN]
Favorite Packages (1)
ZRAP_1234
1
New
Derive New Tree...
New Knowledge Transfer Document
Open
Open in Project Ctrl+Alt+P
Open ABAP Development Object...
Show in ABAP Repository Tree
Copy Ctrl+C
Delete Delete
Remove from Favorite Packages
Get Where-used List Ctrl+Shift+G
Export...
Refresh F5
Share Link...
Activate
Unlock Ctrl+U
Coverage As
Coverage As
Run As
Debug As
Profile As
Expand Folder by...
Configure Tree... Alt+Enter
ABAP Class
ABAP Interface
ABAP Package
2
Other ABAP Repository Object
Outline
There is no active editor


New ABAP Class
ABAP Class
Create an ABAP class
Project: *
TRL_EN
Browse...
Package: *
ZRAP_1234
Browse...
Add to favorite packages
1
Name: *
ZCL_HELLO_WORLD_1234
Description: *
Hello world
Original Language:
EN
Superclass:
Browse...
2
Interfaces:
IF_OO_ADT_CLASSRUN
Add...
Remove
3
< Back
Next >
Finish
Cancel



Implémentation du code

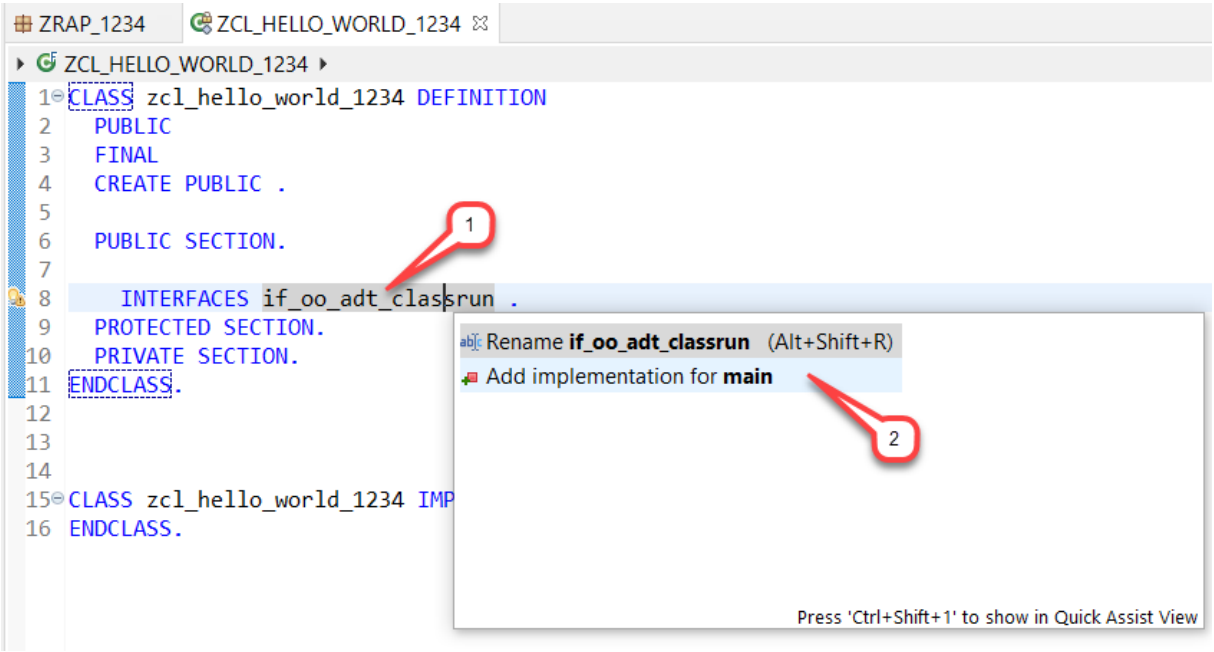
Dans la méthode main, ajoutez ce code :

```
out->write( |Hello world!| ).
```

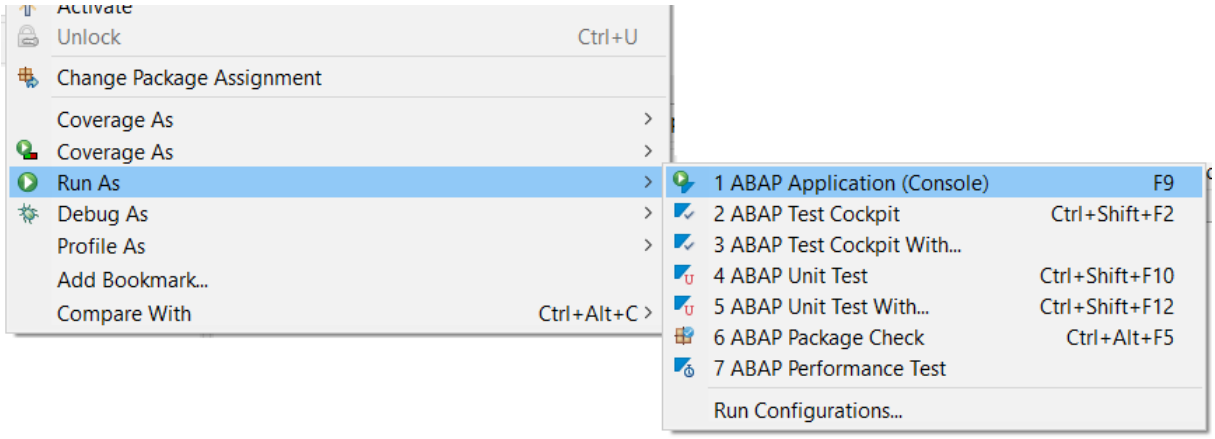
Fonction utilisée : cl_abap_context_info=>get_user_alias( )
Permet de récupérer le nom d'utilisateur dans l'environnement cloud.

Outils utilisés :

- **Ctrl+1** : assistant de correction pour générer main.



- **Ctrl+Space** : complétion automatique du code.
- **Shift+F1** : formatage avec le "Pretty Printer".
- **F9** ou clic droit → *Run As > ABAP Application (Console)* : pour exécuter.



Résultat attendu

- Le message s'affiche dans la console :
  Hello world! (GE200590)
  (ou le nom d'utilisateur réel selon votre système)
- Vous pouvez vider la console via **clic droit** → ***Clear***.

ZRAP_1234 | ZCL_HELLO_WORLD_1234

ZCL_HELLO_WORLD_1234 ▸ IF_OO_ADT_CLASSRUN~MAIN

```
1  CLASS zcl_hello_world_1234 DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5
6    PUBLIC SECTION.
7
8      INTERFACES if_oo_adt_classrun .
9    PROTECTED SECTION.
10   PRIVATE SECTION.
11 ENDCLASS.
12
13
14
15 CLASS zcl_hello_world_1234 IMPLEMENTATION.
16   METHOD if_oo_adt_classrun~main.
17     out->write( |Hello world! ({ cl_abap_context_info=>get_user_alias(  ) })| ).
18   ENDMETHOD.
19
20 ENDCLASS.
```

1

2. Press F9

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Quick Assist | Con

ABAP Console

```
Hello world!

Hello world! (CB0000000021)
```

3

Résumé

À la fin de cette unité, vous avez appris à :

- Créer un package ABAP dans SAP BTP via Eclipse.
- Créer une classe implémentant une interface d'exécution console.
- Utiliser cl_abap_context_info pour obtenir des infos système.
- Exécuter et tester une classe dans ADT (ABAP Development Tools).

3. Semaine 2 – Développement d'une application de rapport de liste en lecture seule

Au cours de cette deuxième semaine, vous allez construire une **application Fiori Elements** basée sur un modèle RAP, capable d'afficher une **liste de voyages (Travel List Report)**. Cette application sera **en lecture seule** (read-only), ce qui signifie qu'aucune modification (création, mise à jour ou suppression) ne sera possible à cette étape du développement.

Vous allez découvrir et manipuler plusieurs composants fondamentaux du modèle RAP, comme les **tables de base de données**, les **CDS views (root & projection)**, les **annotations UI**, et les **services OData exposés automatiquement**.

3.1 Le scénario commercial (Unité 1)

Objectif de l'unité

Cette unité introduit le **contexte métier** de l'application à développer. Il s'agit d'un système de gestion des voyages, basé sur un modèle simple, dans lequel on souhaite afficher une liste de voyages à des fins de consultation uniquement.

Description du scénario

L'application que vous allez développer repose sur le concept de **Business Object Travel**, qui représente un voyage planifié par une agence. Ce BO contient les informations suivantes :

Attribut	**Description**
Travel ID	Identifiant du voyage
Agency ID	Agence qui organise le voyage
Customer ID	Client concerné
Start & End Date	Dates de début et de fin du voyage
Currency / Price	Monnaie utilisée et montant du voyage
Travel Status	Statut (planned, approved, rejected...)

3.2 Création des Tables de Base de Données (Unité 2)

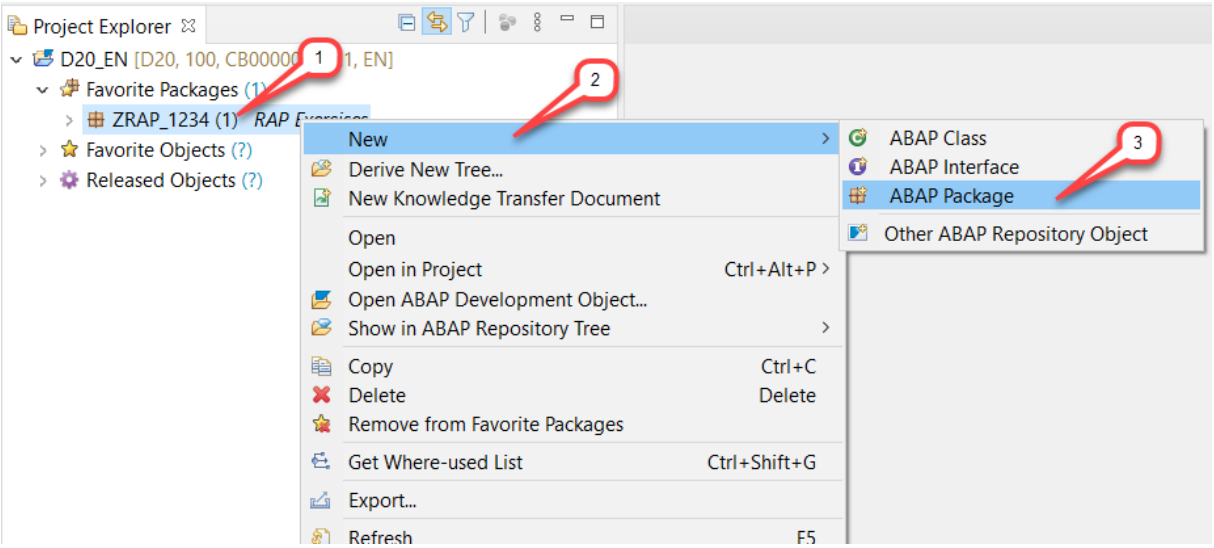
Objectif de l'exercice

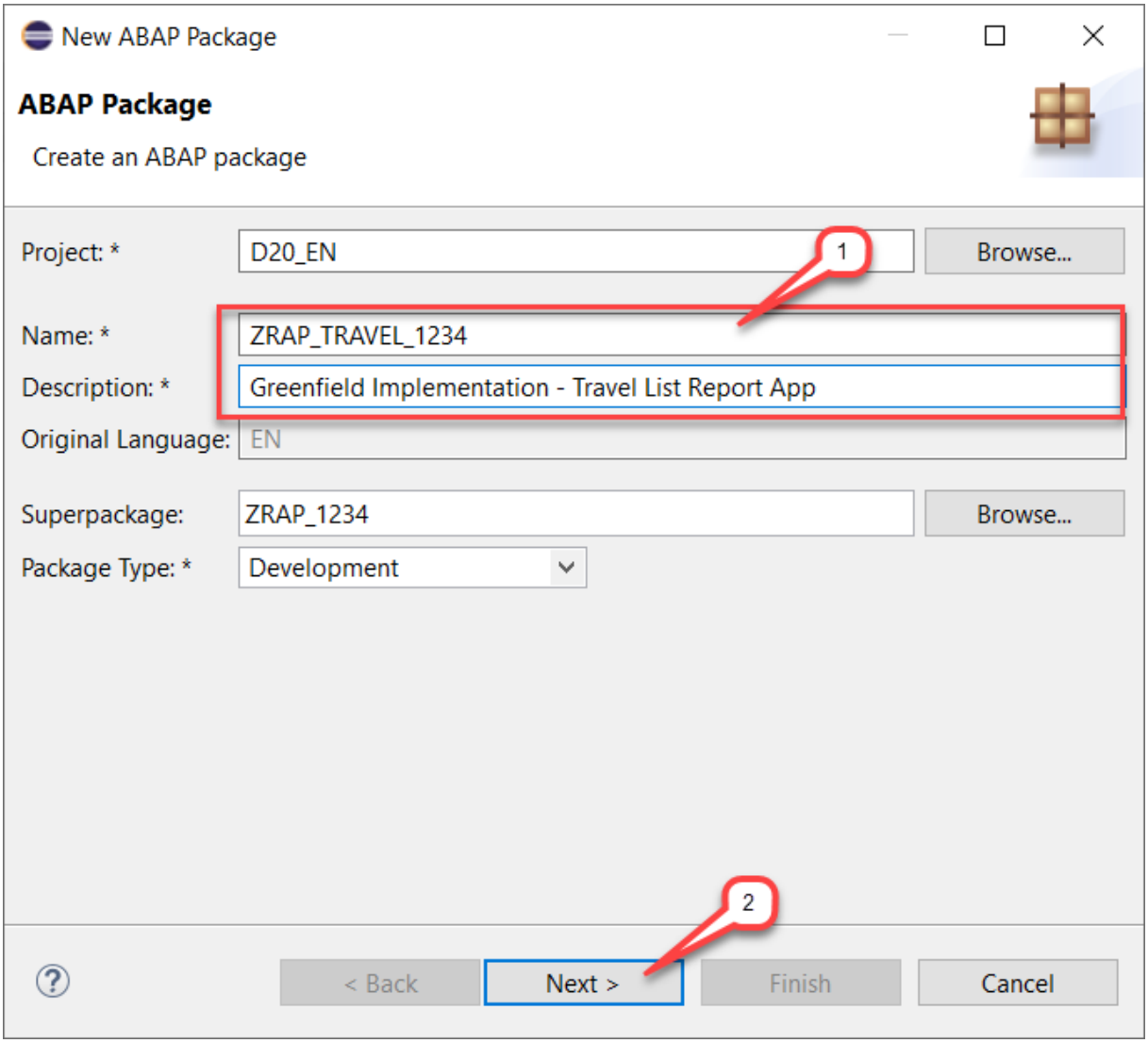
Dans cette unité, vous allez:

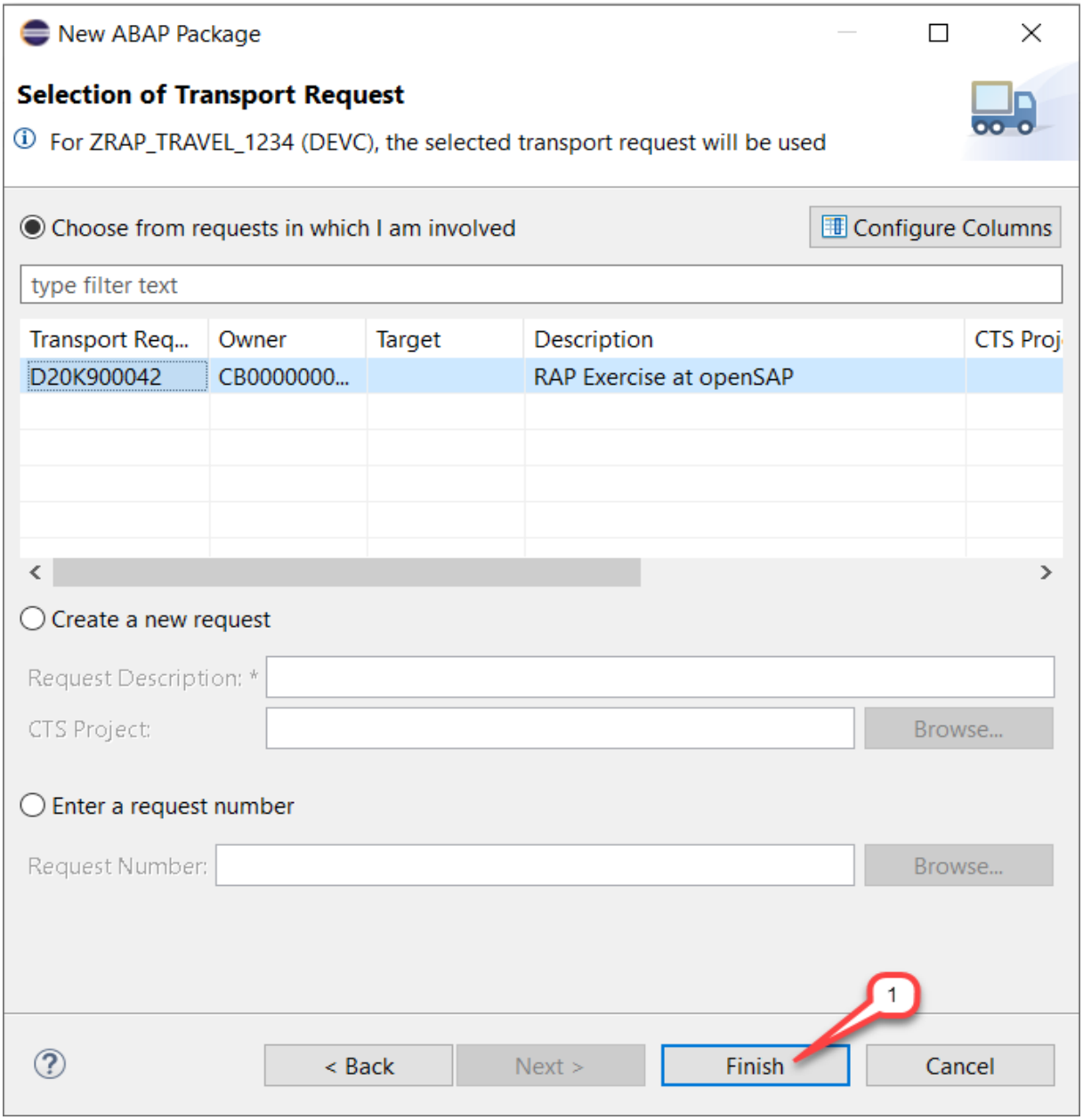
- Créer les **tables physiques** nécessaires à votre application Fiori RAP,
- Organiser votre code dans un **package spécifique**,
- Insérer des **données de démonstration** issues du modèle /DMO/ fourni par SAP.

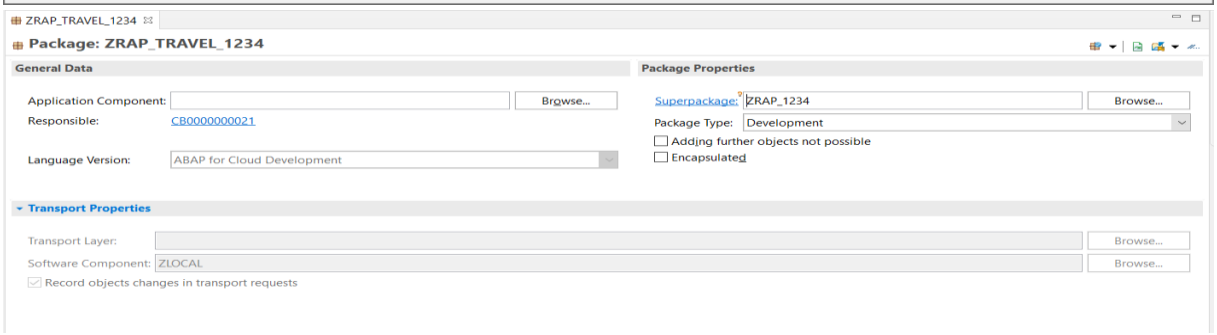
Étape 1 – Créer le package ZRAP_TRAVEL_####

1. Clic droit sur votre package principal ZRAP_#### > **New > ABAP Package**
2. Nom du package : ZRAP_TRAVEL_####
3. Description : *Greenfield Implementation - Travel List Report App*
4. Superpackage : ZRAP_#### (si non prérempli)
5. Sélectionnez ou créez une requête de transport
6. Ajoutez le package aux favoris



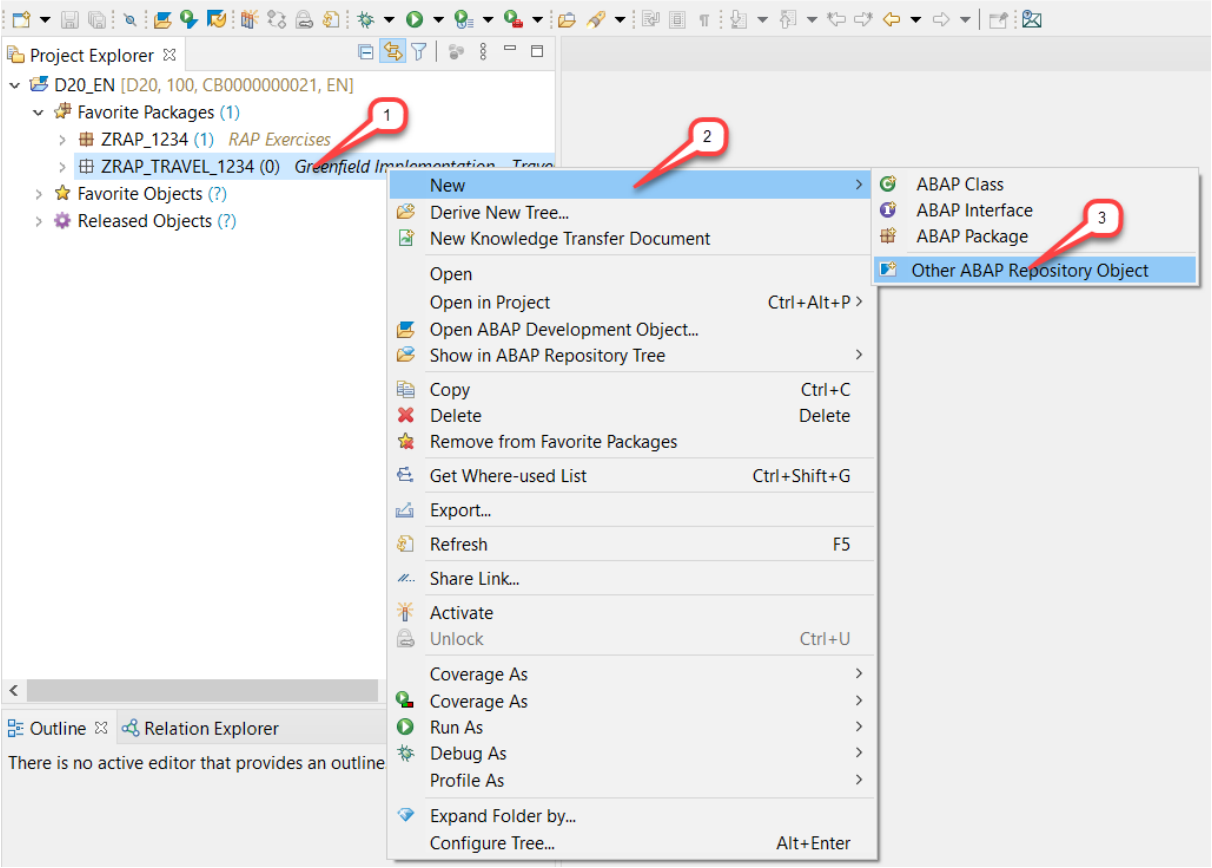

New ABAP Package
ABAP Package
Create an ABAP package
Project: *
D20_EN
Browse...
1
Name: *
ZRAP_TRAVEL_1234
Description: *
Greenfield Implementation - Travel List Report App
Original Language:
EN
Superpackage:
ZRAP_1234
Browse...
Package Type: *
Development
2
< Back
Next >
Finish
Cancel

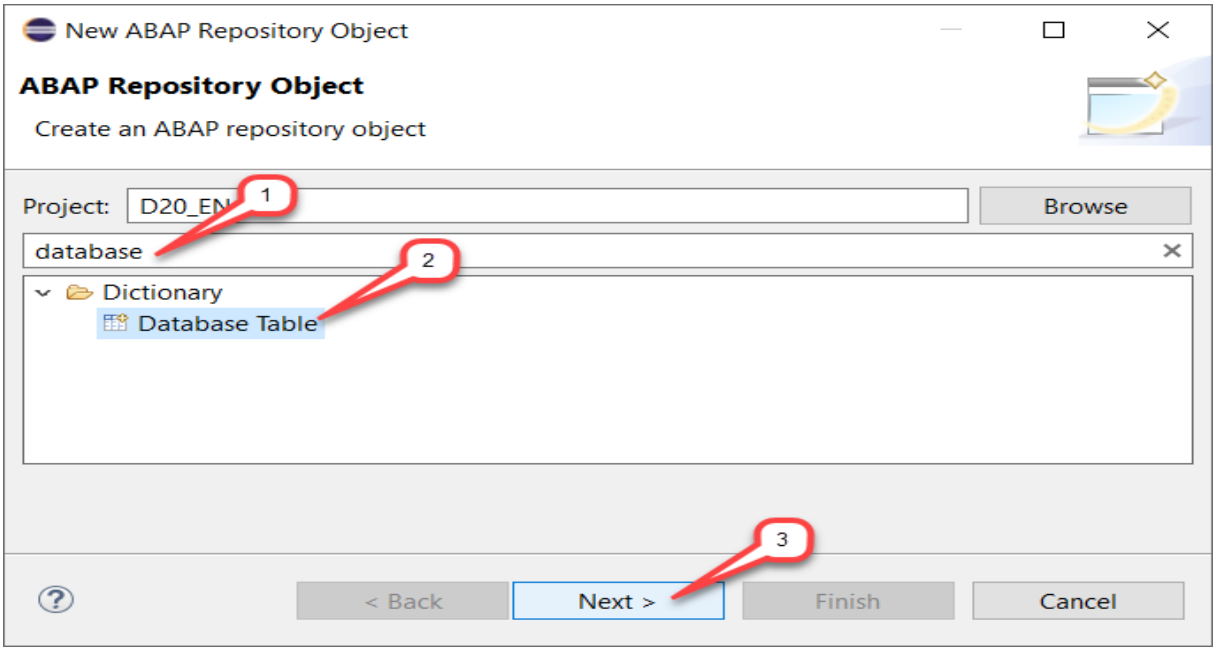

New ABAP Package
Selection of Transport Request
For ZRAP_TRAVEL_1234 (DEVC), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel

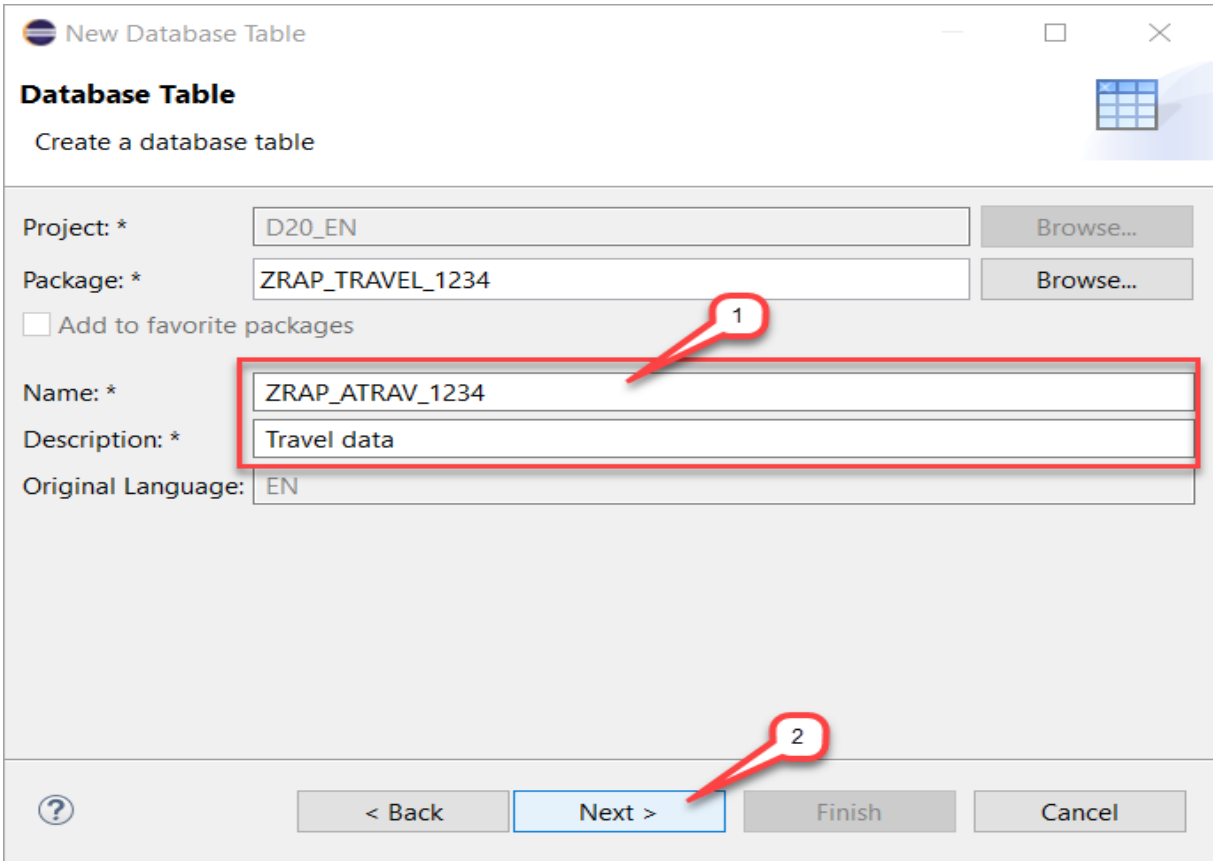

ZRAP_TRAVEL_1234
Package: ZRAP_TRAVEL_1234
General Data
Application Component:
Browse...
Responsible:
CB0000000021
Language Version:
ABAP for Cloud Development
Package Properties
Superpackage:
ZRAP_1234
Browse...
Package Type:
Development
Adding further objects not possible
Encapsulated
Transport Properties
Transport Layer:
Browse...
Software Component:
ZLOCAL
Browse...
Record objects changes in transport requests

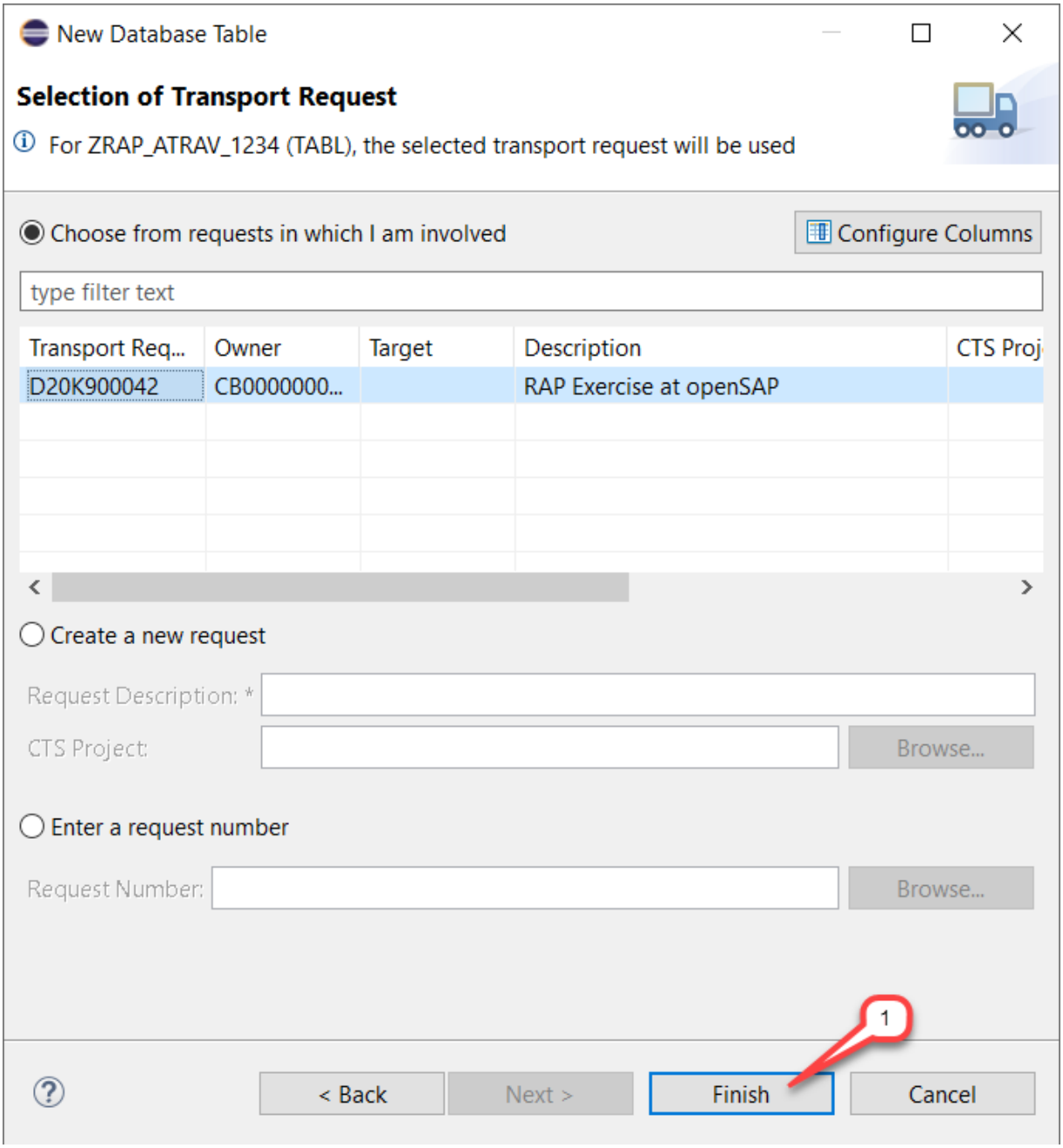
Étape 2 – Créer la table ZRAP_ATRAV_#### (Travel)

1. Clic droit sur le package ZRAP_TRAVEL_#### > **New > Other ABAP Repository Object**
2. Recherchez **Database Table** > **Suivant**
3. Nom : ZRAP_ATRAV_####
4. Description : *Travel data*
5. Assignez la requête de transport




New ABAP Repository Object
ABAP Repository Object
Create an ABAP repository object
Project:
D20_EN
1
Browse
database
2
Dictionary
Database Table
3
< Back
Next >
Finish
Cancel


New Database Table
Database Table
Create a database table
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
1
Name: *
ZRAP_ATRAV_1234
Description: *
Travel data
Original Language:
EN
2
< Back
Next >
Finish
Cancel


New Database Table
Selection of Transport Request
For ZRAP_ATRAV_1234 (TABL), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel

6. Remplacez le code généré par la définition complète de la table

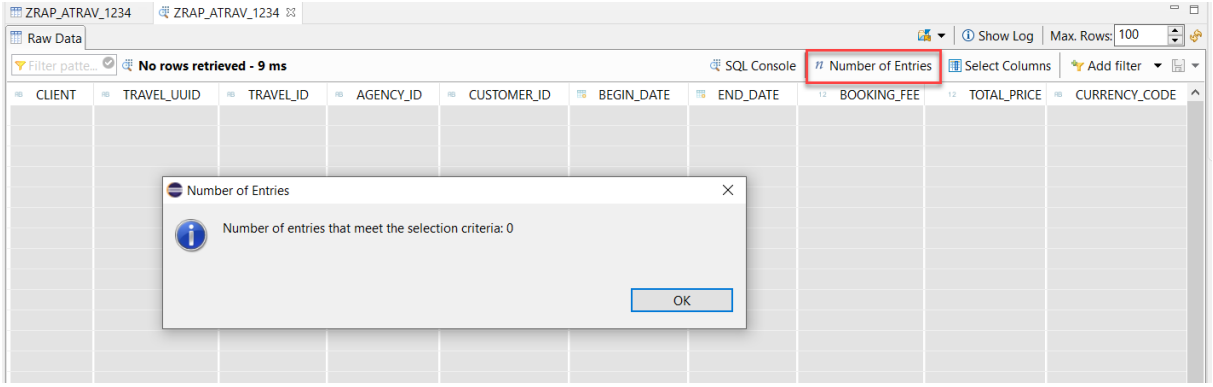
```
@EndUserText.label : 'Travel data'
@AbapCatalog.enhancementCategory : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zrap_atrav_#### {
  key client          : mandt not null;
  key travel_uuid      : sysuuid_x16 not null;
  travel_id           : /dmo/travel_id;
  agency_id           : /dmo/agency_id;
  customer_id         : /dmo/customer_id;
  begin_date          : /dmo/begin_date;
  end_date            : /dmo/end_date;
  @Semantics.amount.currencyCode : 'zrap_atrav_####.currency_code'
  booking_fee         : /dmo/booking_fee;
  @Semantics.amount.currencyCode : 'zrap_atrav_####.currency_code'
  total_price         : /dmo/total_price;
  currency_code       : /dmo/currency_code;
  description         : /dmo/description;
  overall_status      : /dmo/overall_status;
  created_by          : syuname;
  created_at          : timestampl;
```

```
  last_changed_by      : syuname;

  last_changed_at      : timestampl;

  local_last_changed_at : timestampl;

}
```

Brèves explications :

- Certains éléments de données du scénario de référence de vol ABAP (espace de noms **/DMO/**) sont utilisés.
- La clé de la table se compose du **CLIENT**champ et du **TRAVEL_UUID**champ qui est une clé technique (UUID 16 octets).
- Un identifiant de voyage lisible par l'utilisateur :**TRAVEL**_ID
- Le champ **CURRENCY_CODE** est spécifié comme clé de devise pour les champs de montant **BOOKING_FEE** et **TOTAL_PRICE** en utilisant l'annotation sémantique **@Semantics.amount.currencyCode**
- Certains champs administratifs standards sont définis : **CREATED_BY, CREATED_AT**, **LAST_CHANGED_BY**, **LAST_CHANGED_AT** et **LOCAL_LAST_CHANGED_AT**.

7. Enregistrez et activez les modifications.
8. Appuyez sur **F8** pour lancer l'aperçu des données.
   La table de base de données est vide pour le moment ; aucune donnée n'est donc affichée.



Étape 3 – Créer la table ZRAP_ABOOK_#### (Booking)

1. Clic droit sur **Database Tables** > **New > Database Table**

2. Nom : ZRAP_ABOOK_####
3. Description : *Booking data*
4. Assigne la requête de transport
5. Remplacez le code généré par la définition complète de la table

```
@EndUserText.label : 'Booking data'
@AbapCatalog.enhancementCategory : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zrap_abook_#### {
  key client          : mandt not null;
  key booking_uuid     : sysuuid_x16 not null;
  travel_uuid          : sysuuid_x16 not null;
  booking_id           : /dmo/booking_id;
  booking_date         : /dmo/booking_date;
  customer_id          : /dmo/customer_id;
  carrier_id           : /dmo/carrier_id;
  connection_id        : /dmo/connection_id;
  flight_date          : /dmo/flight_date;
  @Semantics.amount.currencyCode : 'zrap_abook_####.currency_code'
  flight_price         : /dmo/flight_price;
  currency_code        : /dmo/currency_code;
  created_by           : syuname;
  last_changed_by      : syuname;
  local_last_changed_at : timestampl;
```

```
}
```

Brèves explications :

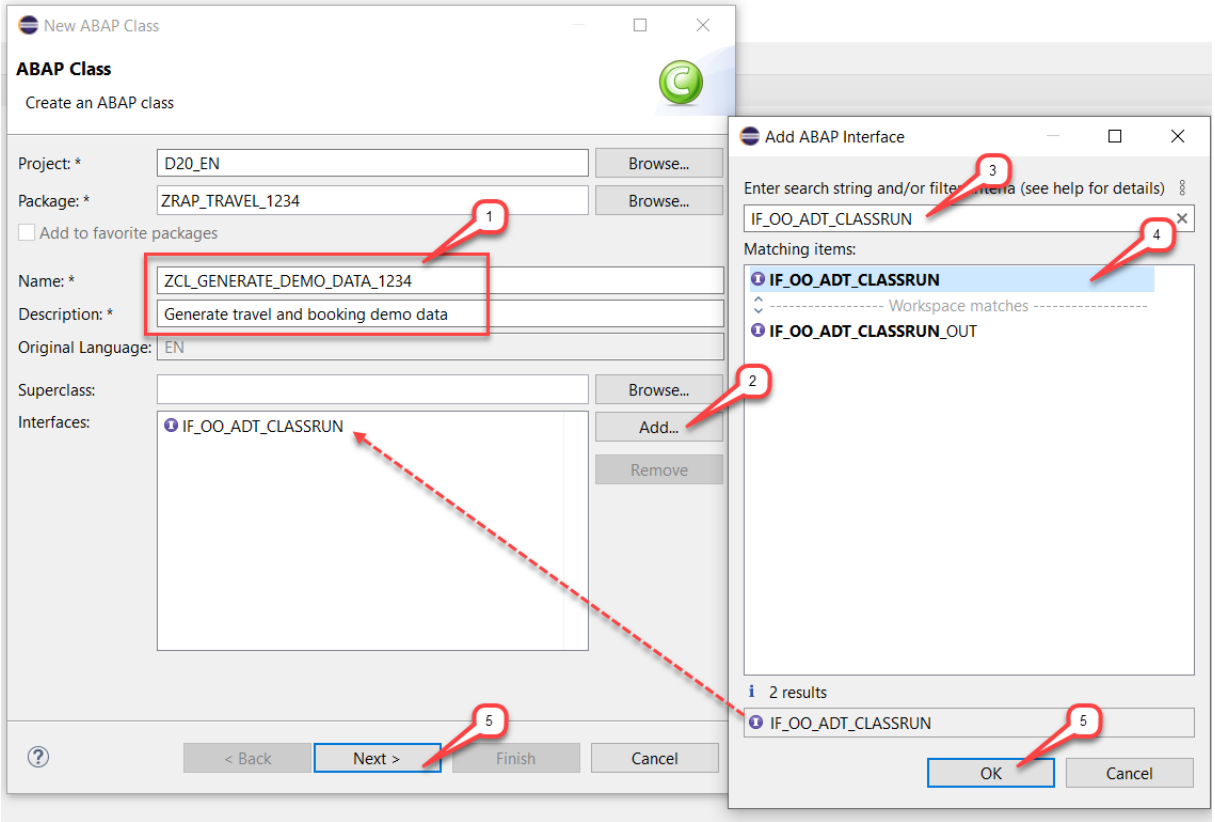
- Certains éléments de données du scénario de référence de vol ABAP (espace de noms **/DMO/**) sont utilisés.
- La clé de la table se compose du **CLIENT** champ et du **BOOKING_UUID** champ qui est une clé technique (UUID 16 octets).
- Un identifiant de voyage lisible par l'utilisateur : **BOOKING_ID**
- Le champ **CURRENCY_CODE** est spécifié comme clé de devise pour le champ de montant **FLIGHT_PRICE** à l'aide de l'annotation sémantique **@Semantics.amount.currencyCode.**
- Certains champs administratifs standards sont définis : **CREATED_BY**, **LAST_CHANGED_BY**, et **LOCAL_LAST_CHANGED_AT**.

6. Enregistrez et activez les modifications.

7. Appuyez sur **F8** pour lancer l'aperçu des données.
   La table de base de données est vide pour le moment ; aucune donnée n'est donc affichée.

Étape 4 – Insérer les données de démonstration

1. Clic droit sur le package ZRAP_TRAVEL_#### > **New > ABAP Class**
2. Nom : ZCL_GENERATE_DEMO_DATA_####
3. Interface : IF_OO_ADT_CLASSRUN
4. Ajoutez une description : *Generate Travel and Booking demo data*
5. Collez l'implémentation de la méthode main fournie




```
METHOD if_oo_adt_classrun~main.


" delete existing entries in the database table

DELETE FROM zrap_atrav_####.

DELETE FROM zrap_abook_####.


" insert travel demo data

INSERT zrap_atrav_#### FROM (

  SELECT

    FROM /dmo/travel

    FIELDS

      uuid( )     AS travel_uuid         ,
```

```
      travel_id     AS travel_id          ,
      agency_id     AS agency_id          ,
      customer_id   AS customer_id        ,
      begin_date    AS begin_date         ,
      end_date      AS end_date           ,
      booking_fee   AS booking_fee        ,
      total_price   AS total_price        ,
      currency_code AS currency_code      ,
      description   AS description        ,
      CASE status
        WHEN 'B' THEN 'A' " accepted
        WHEN 'X' THEN 'X' " cancelled
        ELSE 'O'          " open
      END          AS overall_status      ,
      createdby     AS created_by         ,
      createdat     AS created_at         ,
      lastchangedby AS last_changed_by    ,
      lastchangedat AS last_changed_at    ,
      lastchangedat AS local_last_changed_at
      ORDER BY travel_id UP TO 200 ROWS
  ).
COMMIT WORK.


" insert booking demo data
```

```
INSERT zrap_abook_#### FROM (
  SELECT
    FROM   /dmo/booking    AS booking
      JOIN zrap_atrav_#### AS z
      ON   booking~travel_id = z~travel_id
    FIELDS
      uuid( )              AS booking_uuid        ,
      z~travel_uuid        AS travel_uuid         ,
      booking~booking_id     AS booking_id          ,
      booking~booking_date    AS booking_date        ,
      booking~customer_id     AS customer_id         ,
      booking~carrier_id      AS carrier_id          ,
      booking~connection_id   AS connection_id       ,
      booking~flight_date     AS flight_date         ,
      booking~flight_price    AS flight_price        ,
      booking~currency_code   AS currency_code       ,
      z~created_by         AS created_by          ,
      z~last_changed_by      AS last_changed_by      ,
      z~last_changed_at      AS local_last_changed_by
 ).
COMMIT WORK.


out->write( 'Travel and booking demo data inserted.').
ENDMETHOD.
```

Brèves explications :

- Tout d'abord, toutes les entrées existantes dans les deux tables de base de données sont supprimées.
- Ensuite, les données sont sélectionnées à partir des tables **/DMO/TRAVEL** et **/DMO/BOOKING** insérées dans vos tables ZRAP_ATRAV_####et ZRAP_ABOOK_####respectivement.
- La fonction **SQL UUID( )** est utilisée pour définir la valeur des champs clés **TRAVEL_UUID** et **BOOKING_UUID**.
- L'instruction **COMMIT WORK** est ensuite exécutée pour conserver les données. La sélection de données a été limitée à 200 enregistrements de voyage, mais vous pouvez modifier ce nombre si vous le souhaitez.
- Un message de réussite est écrit dans la console à la fin.

6. Enregistrez et activez les modifications.
7. Appuyez sur **F9** pour exécuter la classe ABAP en tant qu'application console pour générer les données de démonstration et remplir vos tables.

Étape 5 – Exécuter la classe et vérifier les données

1. Clique droit sur la classe > **Run As > ABAP Application (Console)**
2. Vérifiez le message affiché dans la console
3. Vous pouvez maintenant prévisualiser les données des tables de base de données Voyages et Réservations.
   Sélectionnez la table de base de données concernée
   (ZRAP_ATRAV_####ou ZRAP_ABOOK_####le suffixe ####) dans l'Explorateur de projets et appuyez sur **F8** .
4. L'aperçu des données s'ouvre dans l'éditeur.
   Vous pouvez l'utiliser pour modifier les paramètres, par exemple en sélectionnant le **nombre d'entrées** ou **la console SQL** , ou en filtrant les entrées avec **« Ajouter un filtre »** .
5. Ouvre la **Data Preview** des tables ZRAP_ATRAV_#### et ZRAP_ABOOK_####
6. Explore les données (filtres, SQL console, nombre d'entrées…)

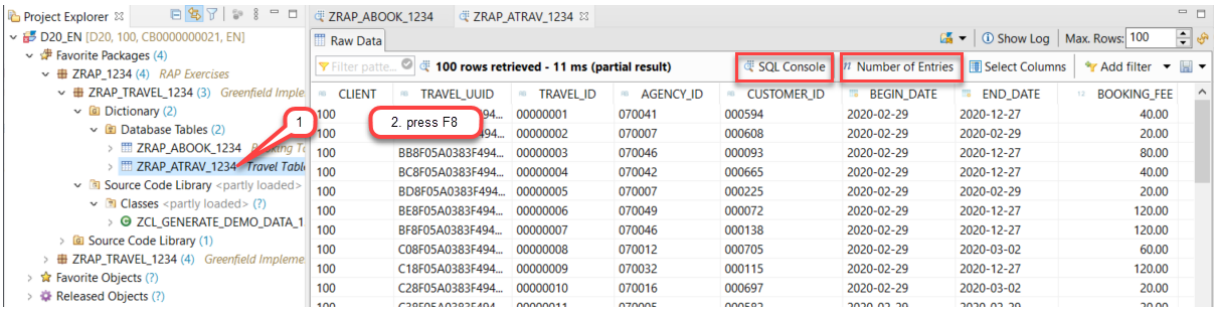
```
69      COMMIT WORK.
70
71      out->write( 'Travel and booking demo data inserted.').
72    ENDMETHOD.
73
74  ENDCLASS.
```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Console

ABAP Console

```
Travel and booking demo data inserted.
```



Résumé

Dans cette unité, vous avez appris à :

- Organiser tes artefacts de développement dans un package dédié,
- Créer deux tables relationnelles pour les entités Travel et Booking,
- Remplir les tables avec des **données de test réalistes** issues du modèle de démonstration /DMO/,
- Utiliser ADT pour **exécuter, activer et visualiser** des objets ABAP.

3.3 Création du modèle de données CDS (Core Data Service) (Unité 3)

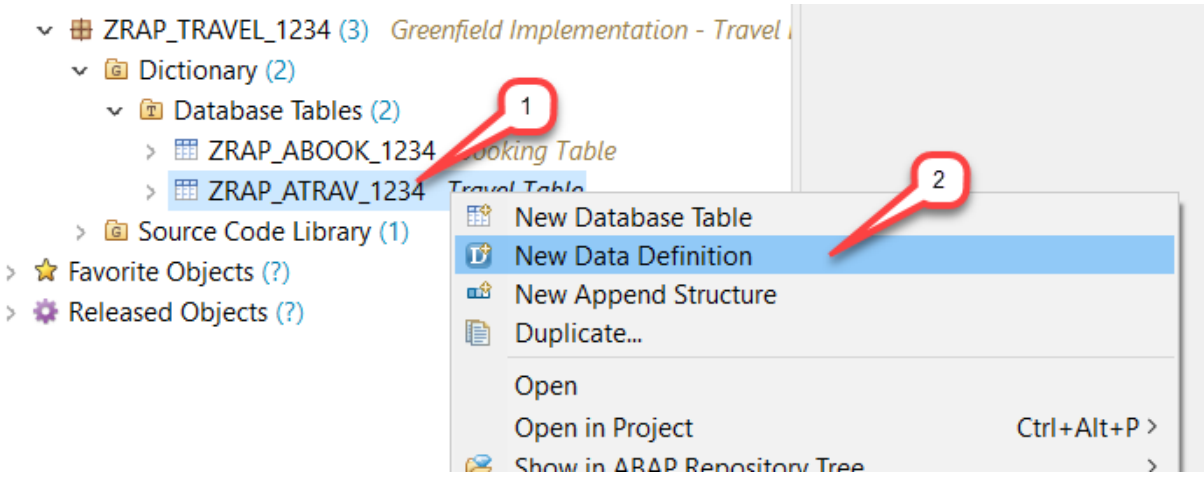
Objectif de l'unité

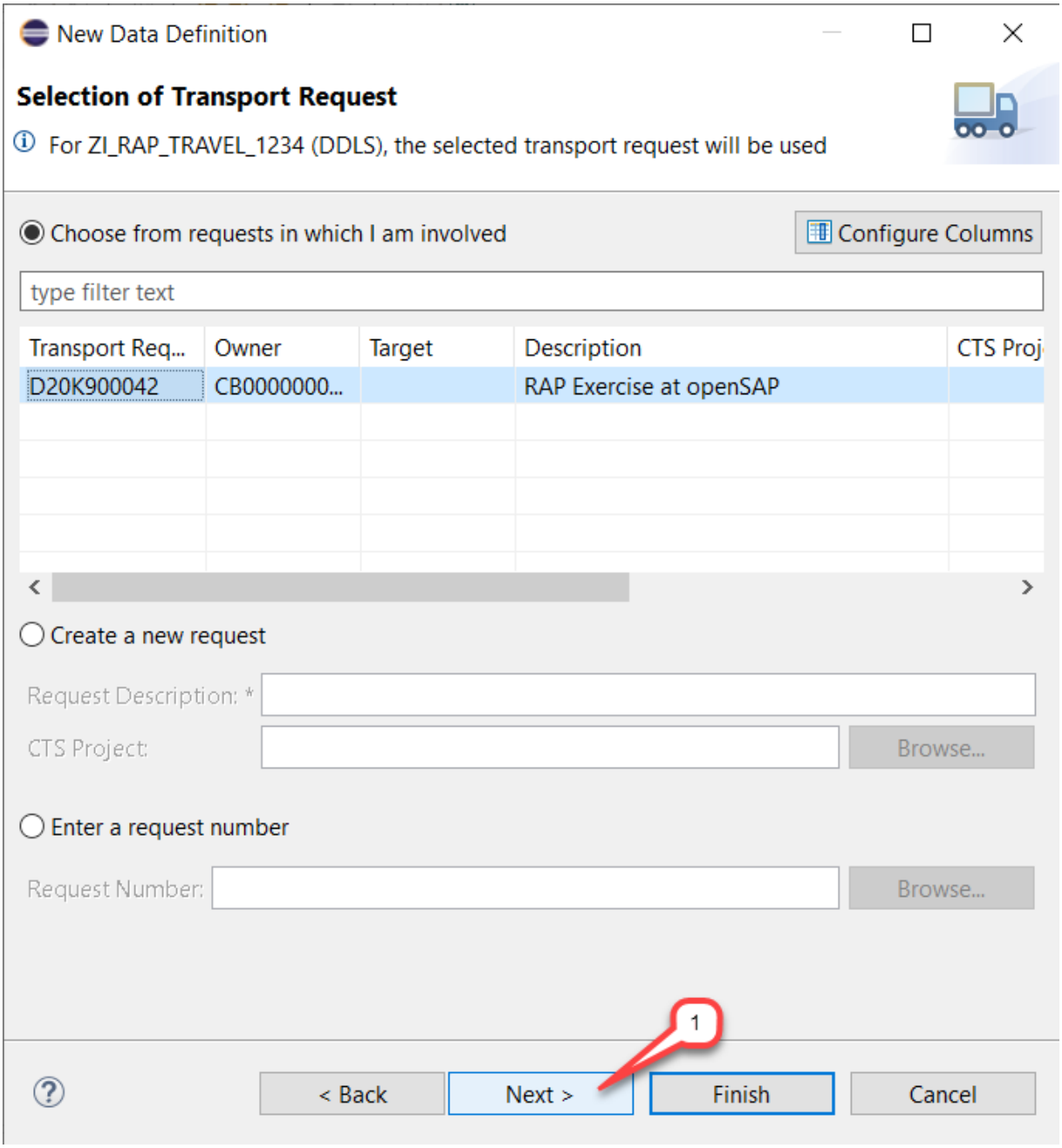
Dans cette unité, vous allez:

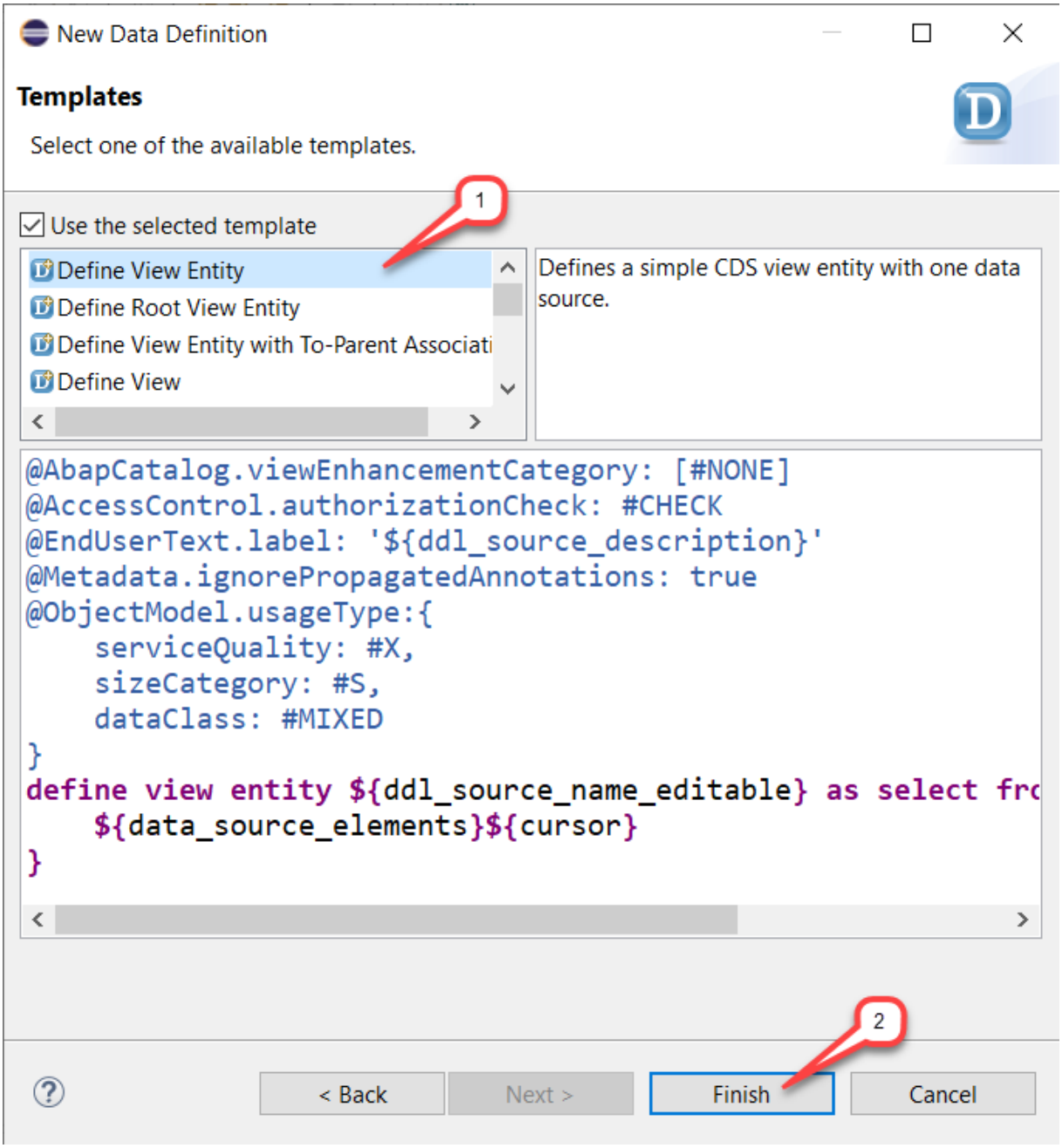
- Créer les vues **CDS Interface** (Business Object views) pour les entités Travel et Booking,
- Définir les **associations** avec d'autres entités (Client, Agence, Vol, etc.),
- Construire la **structure de composition** du Business Object (Travel = racine, Booking = enfant).

Étape 1 – Créer la vue interface CDS ZI_RAP_Travel_####

1. Clic droit sur la table ZRAP_ATRAV_#### > **New Data Definition**
2. Nom : ZI_RAP_Travel_####
3. Description : *Vue BO Voyage*
4. Choisissez le modèle **Define View Entity**
5. Affectez une requête de transport




New Data Definition
Selection of Transport Request
For ZI_RAP_TRAVEL_1234 (DDLS), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel


New Data Definition
Templates
Select one of the available templates.
Use the selected template
1
Define View Entity
Define Root View Entity
Define View Entity with To-Parent Associati
Define View
Defines a simple CDS view entity with one data source.
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: '${ddl_source_description}'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
serviceQuality: #X,
sizeCategory: #S,
dataClass: #MIXED
}
define view entity ${ddl_source_name_editable} as select fro
${data_source_elements}${cursor}
}
2
< Back
Next >
Finish
Cancel

```
@AccessControl.authorizationCheck : #CHECK

 @EndUserText.label : « Vue BO de voyage »

 définir l'entité de vue ZI_RAP_Travel_####

  comme sélectionné dans zrap_atrav_#### comme Voyage



  association [0..*] à ZI_RAP_Booking_#### comme _Booking sur $projection.TravelUUID =
_Booking.TravelUUID



  association [0..1] à /DMO/I_Agency comme _Agency sur $projection.AgencyID = _Agency.AgencyID

  association [0..1] à /DMO/I_Customer en tant que _Customer sur $projection.CustomerID =
_Customer.CustomerID

  association [0..1] à I_Currency comme _Currency sur $projection.CurrencyCode =
_Currency.Currency

 {

  clé travel_uuid comme TravelUUID,

      travel_id comme TravelID,

      agency_id comme AgencyID,

      customer_id comme CustomerID,

      begin_date comme BeginDate,

      end_date comme EndDate,

      @Semantics.amount.currencyCode : « CurrencyCode »

      booking_fee comme BookingFee,

      @Semantics.amount.currencyCode : « CurrencyCode »

      total_price comme PrixTotal,

      currency_code comme CurrencyCode,

      description comme Description,
```

```
    overall_status comme TravelStatus,
    @Semantics.user.createdBy : vrai
    created_by comme CréePar,
    @Semantics.systemDateTime.createdAt : vrai
    created_at comme CreatedAt,
    @Semantics.user.lastChangedBy : vrai
    last_changed_by comme LastChangedBy,
    @Semantics.systemDateTime.lastChangedAt : vrai
    last_changed_at comme LastChangedAt,
    @Semantics.systemDateTime.localInstanceLastChangedAt : vrai
    local_last_changed_at comme LocalLastChangedAt,

    /* associations */
    _Réservation,
    _Agence,
    _Client,
    _Devise
}
```

⚠️ **Ne pas activer** la vue immédiatement (car ZI_RAP_Booking_#### n'existe pas encore).

Étape 2 – Créer la vue interface CDS ZI_RAP_Booking_####

1. Clic droit sur la table ZRAP_ABOOK_#### > **New Data Definition**
2. Nom : ZI_RAP_Booking_####
3. Description : *Vue BO Réservation*

4. Choisissez le modèle **Define View Entity**
5. Affectez une requête de transport

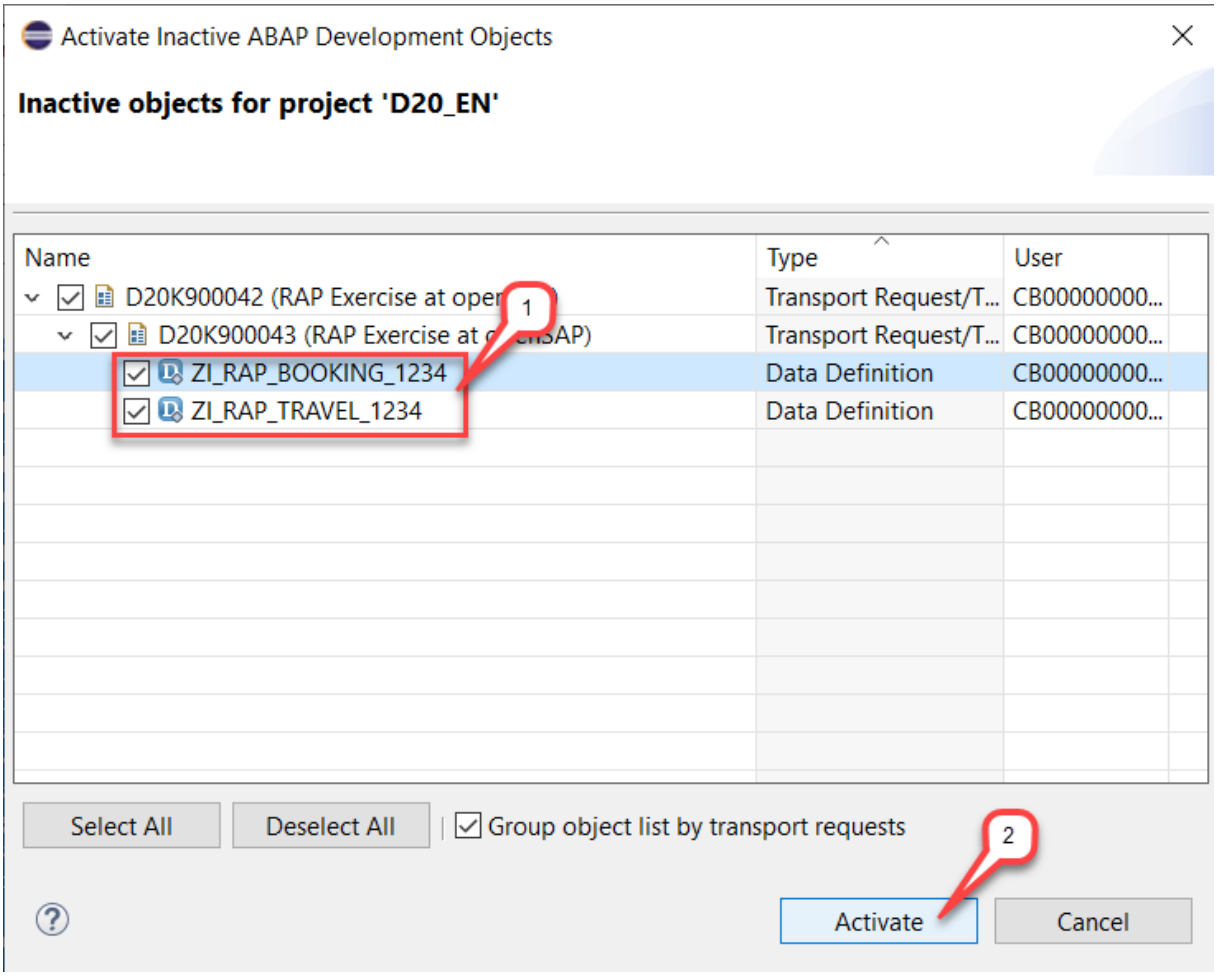
```
@AccessControl.authorizationCheck : #CHECK

 @EndUserText.label : « Vue BO de réservation »

 définir l'entité de vue ZI_RAP_Booking_####

  comme sélectionné dans zrap_abook_#### comme Réservation



  association [1..1] à ZI_RAP_Travel_#### comme _Travel sur $projection.TravelUUID =
_Travel.TravelUUID



  association [1..1] à /DMO/I_Customer en tant que _Customer sur $projection.CustomerID =
_Customer.CustomerID

  association [1..1] à /DMO/I_Carrier comme _Carrier sur $projection.CarrierID = _Carrier.AirlineID

  association [1..1] à /DMO/I_Connection comme _Connection sur $projection.CarrierID =
_Connection.AirlineID

                                        et $projection.ConnectionID = _Connection.ConnectionID

  association [1..1] à /DMO/I_Flight comme _Flight sur $projection.CarrierID = _Flight.AirlineID

                                        et $projection.ConnectionID = _Flight.ConnectionID

                                        et $projection.FlightDate = _Flight.FlightDate

  association [0..1] à I_Currency comme _Currency sur $projection.CurrencyCode =
_Currency.Currency

 {

  clé booking_uuid comme BookingUUID,

     travel_uuid comme TravelUUID,

     booking_id comme BookingID,

     booking_date comme BookingDate,
```

```
    customer_id comme CustomerID,
    carrier_id comme CarrierID,
    connection_id comme ConnectionID,
    flight_date comme FlightDate,
    @Semantics.amount.currencyCode : « CurrencyCode »
    flight_price comme FlightPrice,
    currency_code comme CurrencyCode,
    @Semantics.user.createdBy : vrai
    created_by comme CréePar,
    @Semantics.user.lastChangedBy : vrai
    last_changed_by comme LastChangedBy,
    @Semantics.systemDateTime.localInstanceLastChangedAt : vrai
    local_last_changed_at comme LocalLastChangedAt,

    /* associations */
    _Voyage,
    _Client,
    _Transporteur,
    _Connexion,
    _Vol,
    _Devise
}
```

⚠️ **Ne pas activer** immédiatement (il faut activer les 2 vues en même temps à l'étape suivante).

Étape 3 – Activer les deux vues CDS ensemble

1. Allerz dans Eclipse > ZI_RAP_Travel_#### et ZI_RAP_Booking_####
2. Menu **Activate All** ou **Ctrl + Shift + F3**
3. Sélectionnez les deux définitions > **Activate**

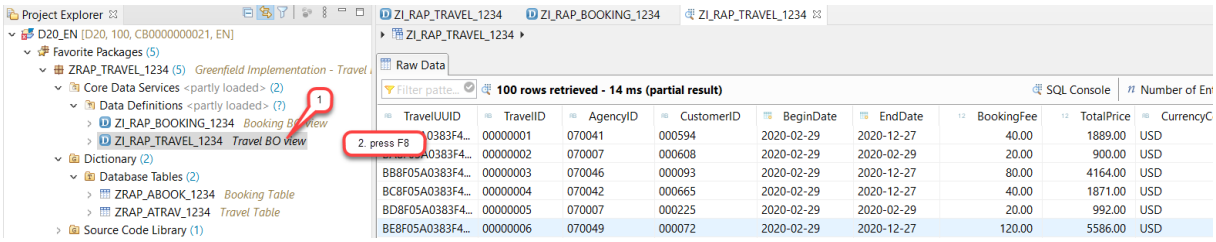


Les **warnings** concernant @AccessControl.authorizationCheck : #CHECK sont normaux (pas encore de DCL définis).

Étape 4 – Prévisualiser les données

1. Sélectionne une des vues (ZI_RAP_Travel_#### ou ZI_RAP_Booking_####)

2. Appuie sur **F8** pour afficher l'aperçu

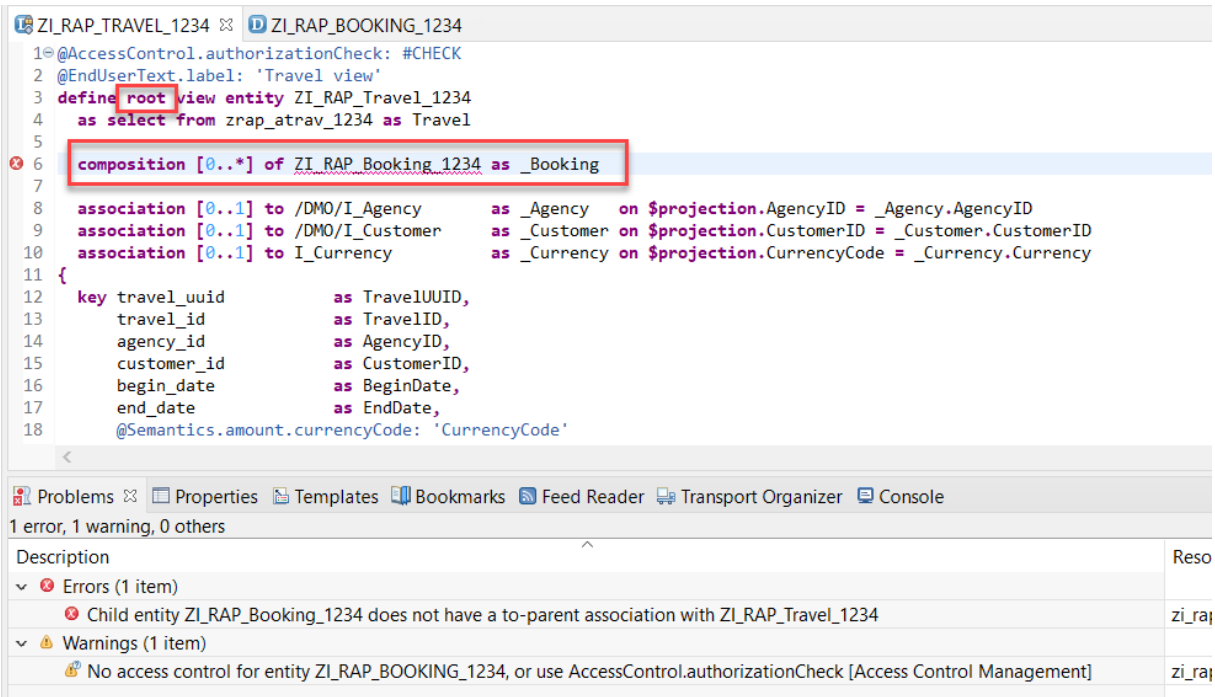


Étape 5 – Définir la structure de composition (modèle BO)

Dans la vue ZI_RAP_Travel_#### :

- Ajouter **root** à la ligne define root view entity
- Modifier l'association _Booking en :

composition [0..*] de ZI_RAP_Booking_#### comme _Booking



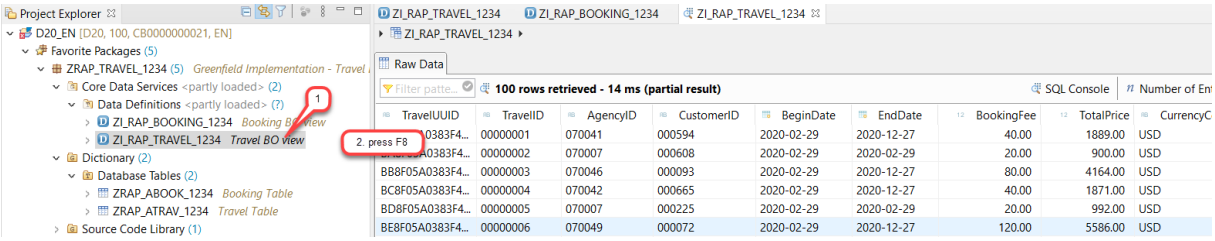
Dans la vue ZI_RAP_Booking_#### :

- Remplacer l'association _Travel par :

association au parent ZI_RAP_Travel_#### comme _Travel sur $projection.TravelUUID = _Travel.TravelUUID

Activer de nouveau les deux vues ensemble

1. Ctrl + Shift + F3 > Sélectionne ZI_RAP_Travel_#### et ZI_RAP_Booking_####
2. Cliquez sur **Activer**



Résumé de l'unité

Dans cette unité, vous avez appris :

- Ce qu'est un modèle de données CDS (CDS Data Model)
- À créer des vues interface (ZI_RAP_...) avec des associations riches
- À préparer un modèle BO avec **arbre de composition** (root, composition, association to parent)
- À utiliser les annotations @Semantics pour les champs monétaires et les données administratives
- À activer et tester tes vues avec **Data Preview**

3.4 Création d'une projection de modèle de données CDS (Unité 4)

Objectif de l'unité

Dans cette unité, tu vas :

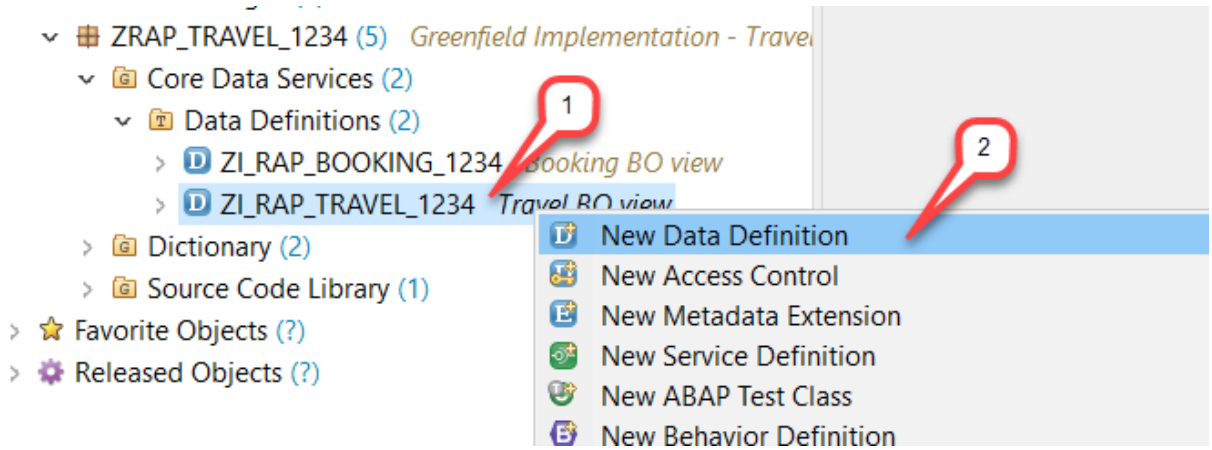
- Créer une **CDS Projection View** ZC_RAP_Travel_####,
- Reposer sur la **CDS Interface View** ZI_RAP_Travel_####,
- Sélectionner uniquement les éléments utiles à votre application (liste de voyages),

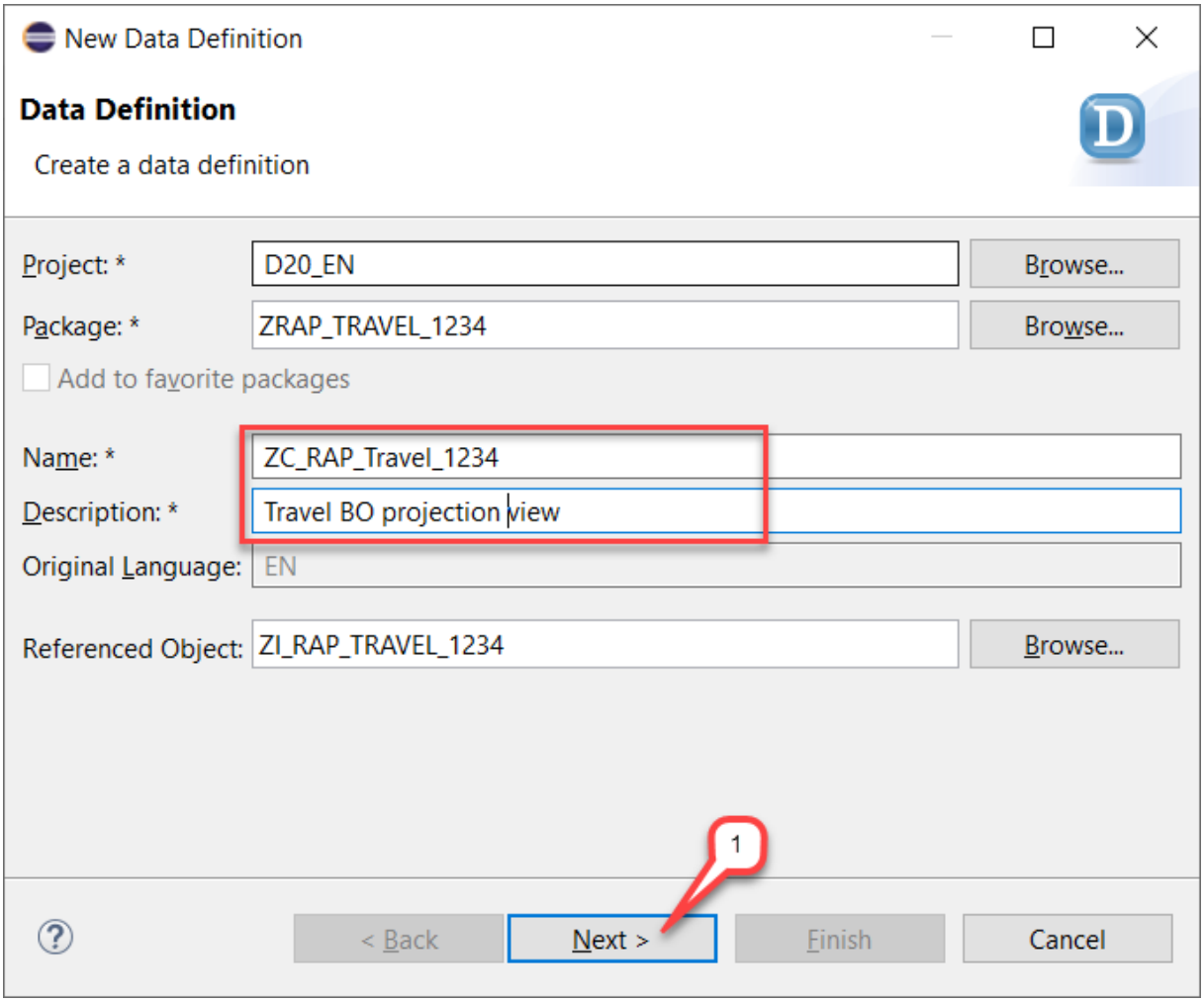
- Ajouter des annotations **UI**, **recherche**, et **sémantiques**.

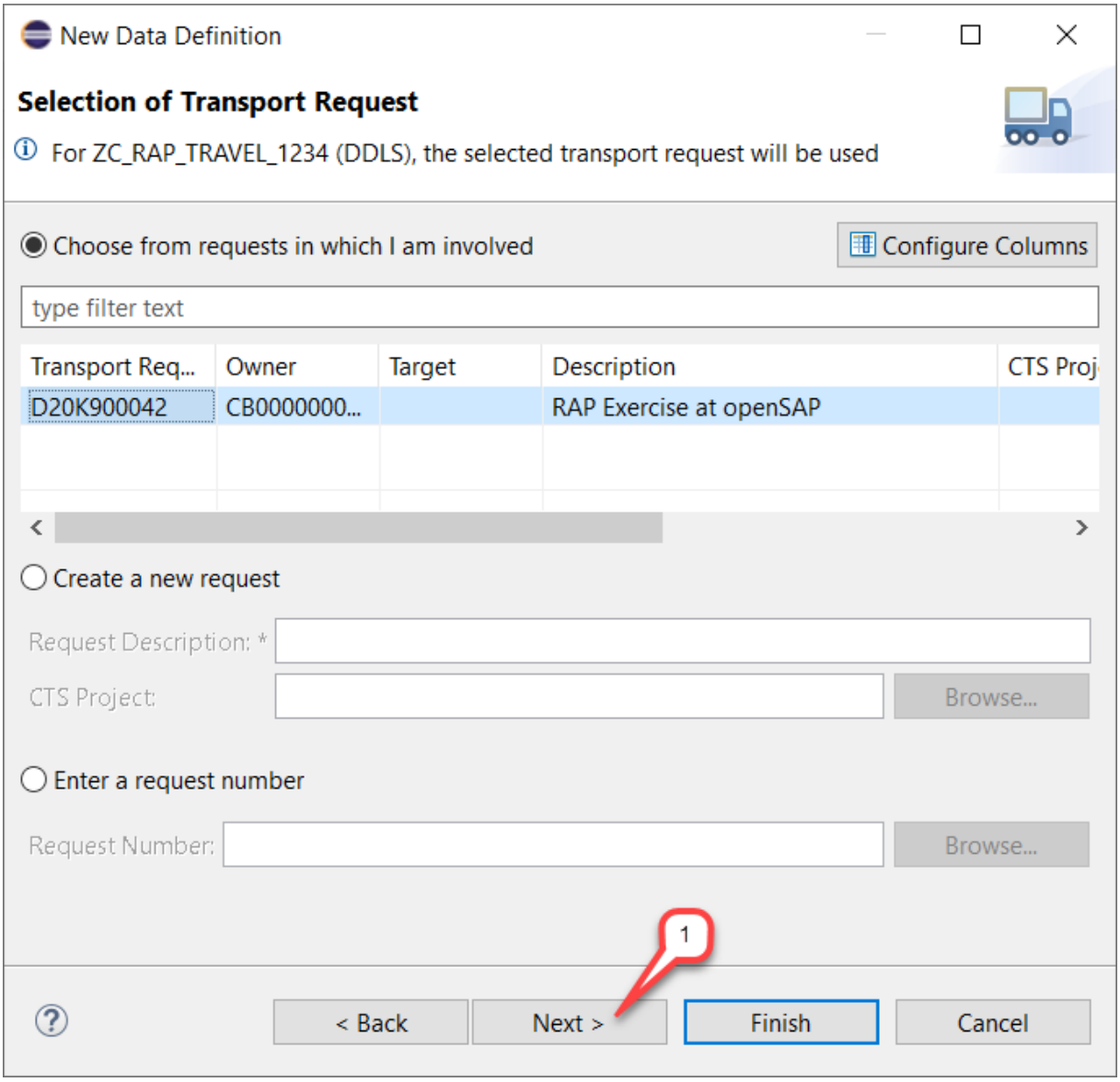
Étape 1 – Création de la vue de projection de voyage : ZC_RAP_Travel_####

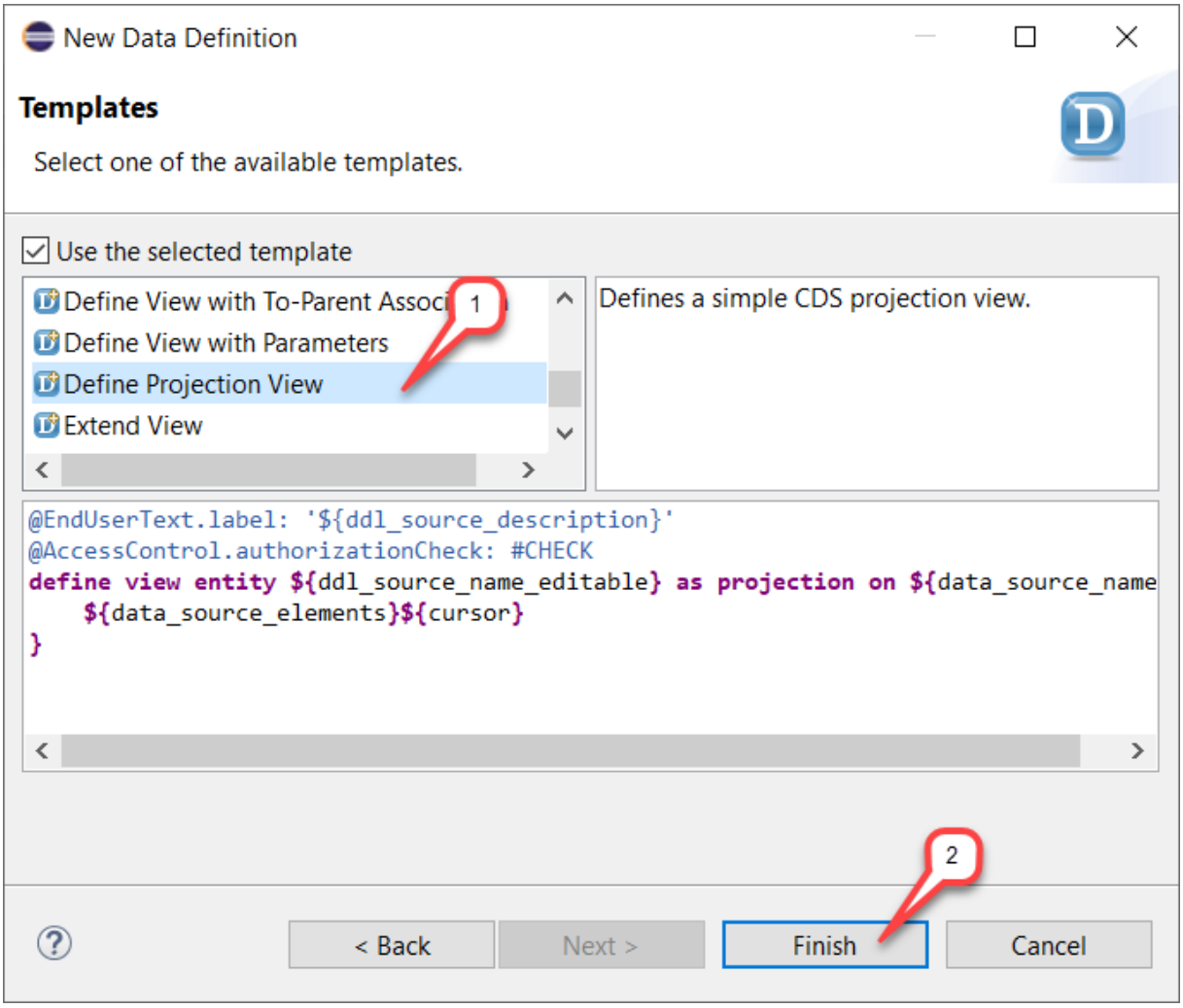
1. Clic droit sur ZI_RAP_Travel_#### > **New Data Definition**
2. Nom : ZC_RAP_Travel_####
3. Description : *Vue de projection Travel BO*
4. Type de modèle : Define Root View Entity as Projection On

Transport > **Suivant** > **Terminer**




New Data Definition
Data Definition
Create a data definition
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
Name: *
ZC_RAP_Travel_1234
Description: *
Travel BO projection view
Original Language:
EN
Referenced Object:
ZI_RAP_TRAVEL_1234
Browse...
1
< Back
Next >
Finish
Cancel


New Data Definition
Selection of Transport Request
For ZC_RAP_TRAVEL_1234 (DDLS), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel



Rappels :

- ZI_ → Interface View (BO technique)
- ZC_ → Consumption View (projection vers UI)
- La projection est basée sur ZI_RAP_Travel_####
- Le mot-clé **redirected to composition child** lie vers la vue de projection ZC_RAP_Booking_####.

Code projeté

```
@EndUserText.label : « Vue de projection BO de voyage »

@AccessControl.authorizationCheck : #CHECK

@Search.searchable : vrai

@Metadata.allowExtensions : vrai


définir l'entité de vue racine ZC_RAP_Travel_####

  en projection sur ZI_RAP_Travel_#### en tant que Voyage

{

  clé TravelUUID,

      @Search.defaultSearchElement : vrai

      Carte d'identité de voyage,

      @Consumption.valueHelpDefinition : [{ entité : { nom : '/DMO/I_Agency', élément : 'AgencyID'} }]

      @ObjectModel.text.element : ['AgencyName']

      @Search.defaultSearchElement : vrai

      ID d'agence,

      _Agency.Name comme AgencyName,

      @Consumption.valueHelpDefinition : [{ entité : { nom : '/DMO/I_Customer', élément :
'CustomerID'} }]

      @ObjectModel.text.element : ['CustomerName']

      @Search.defaultSearchElement : vrai

      ID client,

      _Customer.LastName comme CustomerName,

      Date de début,

      Date de fin,

      @Semantics.amount.currencyCode : « CurrencyCode »
```

```
    Frais de réservation,

    @Semantics.amount.currencyCode : « CurrencyCode »

    Prix total,

    @Consumption.valueHelpDefinition : [{ entité : { nom : 'I_Currency', élément : 'Currency'} }]

    Code de devise,

    Description,

    Statut de voyage,

    Dernière modification à,

    LocalLastChangedAt,


    /* Associations */

    _Agence,

    _Réservation : redirigé vers la composition enfant ZC_RAP_Booking_####,

    _Devise,

    _Client

}
```

Enregistrez les modifications, mais **n'activez PAS** encore la vue de projection BO de voyage.

Étape 2 – Création de la vue de projection de réservation : ZC_RAP_Booking_####

1. Clic droit sur ZI_RAP_Booking_#### > **New Data Definition**
2. Nom : ZC_RAP_Booking_####
3. Description : *Vue de projection Booking BO*
4. Type : Define Projection View
5. **Transport** > **Suivant** > **Terminer**

Code projeté

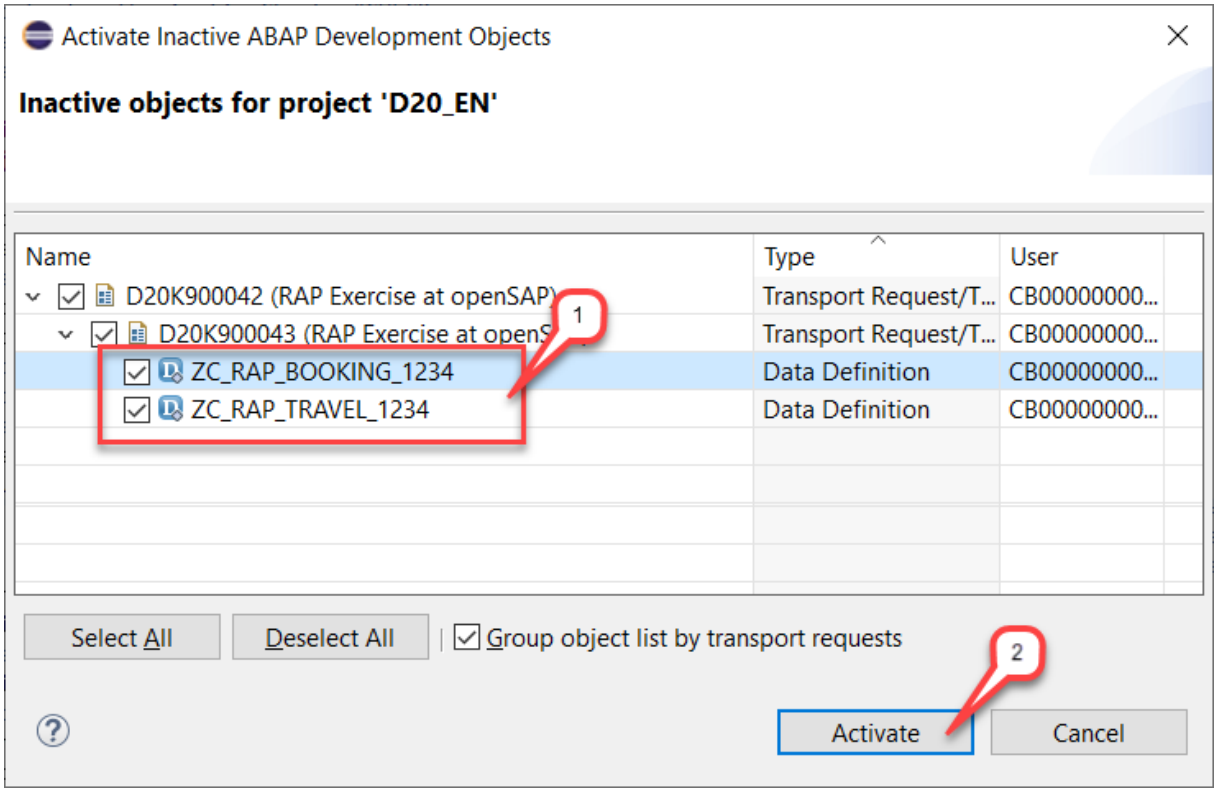
```
@EndUserText.label : « Vue de projection BO de réservation »

 @AccessControl.authorizationCheck : #CHECK

 @Search.searchable : vrai

 @Metadata.allowExtensions : vrai



 définir l'entité de vue ZC_RAP_Booking_####

  en projection sur ZI_RAP_Booking_#### en tant que réservation

{

  clé BookingUUID,

      TravelUUID,

      @Search.defaultSearchElement : vrai

      ID de réservation,

      Date de réservation,

      @Consumption.valueHelpDefinition : [{ entité : {nom : '/DMO/I_Customer', élément :
'CustomerID' } }]

      @ObjectModel.text.element : ['CustomerName']

      @Search.defaultSearchElement : vrai

      ID client,

      _Customer.LastName comme CustomerName,

      @Consumption.valueHelpDefinition : [{entité : {nom : '/DMO/I_Carrier', élément : 'AirlineID' }}]

      @ObjectModel.text.element : ['CarrierName']

      ID de l'opérateur,

      _Carrier.Name comme CarrierName,
```

```
    @Consumption.valueHelpDefinition : [ {entité : {nom : '/DMO/I_Flight', élément :
'ConnectionID'},
                                additionalBinding : [ { localElement : 'CarrierID', element : 'AirlineID' },
                                        { localElement: 'FlightDate', élément: 'FlightDate', utilisation:
#RESULT},
                                        { localElement: 'FlightPrice', element: 'Price', usage: #RESULT },
                                        { localElement: 'CurrencyCode', élément: 'CurrencyCode',
utilisation: #RESULT } ] } ]
    ID de connexion,
    Date du vol,
    @Semantics.amount.currencyCode : « CurrencyCode »
    Prix du vol,
    @Consumption.valueHelpDefinition : [{entité : {nom : 'I_Currency', élément : 'Currency' }}]
    Code de devise,
    LocalLastChangedAt,

    /* associations */
    _Voyage : redirigé vers le parent ZC_RAP_Travel_####,
    _Client,
    _Transporteur,
    _Connexion,
    _Vol
}
```

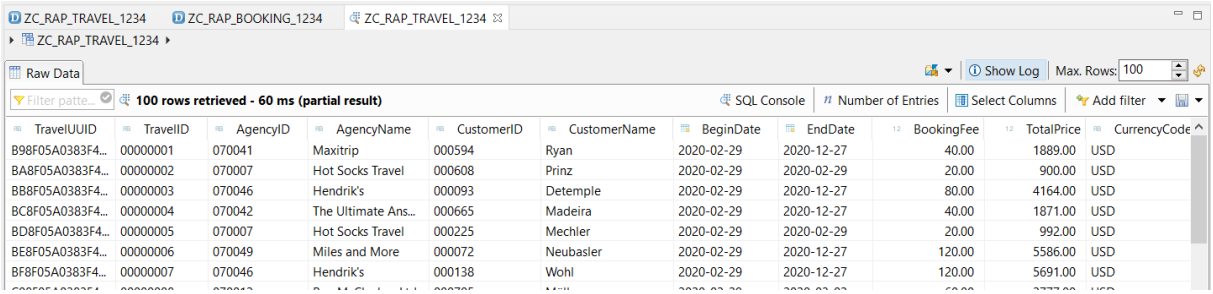
Enregistrez les modifications, mais **n'activez PAS** encore la vue de projection BO de réservation.

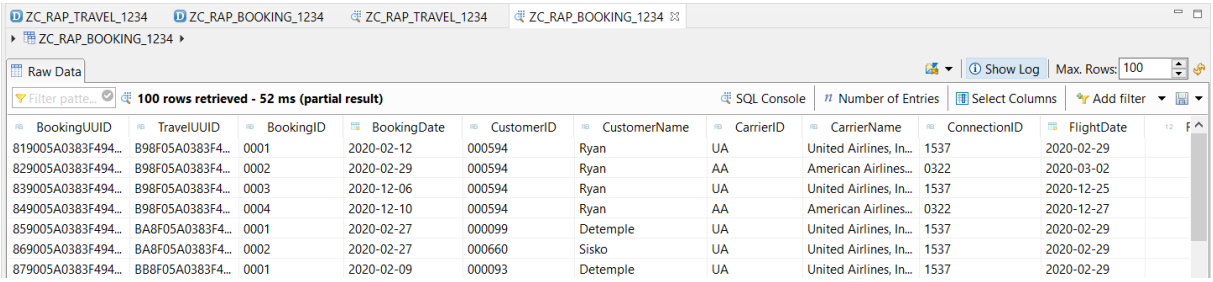
Étape 3 – Activation et aperçu des données

1. Activez les deux vues ZC_RAP_Travel_#### et ZC_RAP_Booking_#### ensemble **(Ctrl+Maj+F3)**



2. Lancez l'aperçu **(F8)** pour vérifier les données





3. Des **warnings** peuvent apparaître à cause de @AccessControl.authorizationCheck: #CHECK sans définition de **CDS Access Control**

Résumé de l'unité

Dans cette unité, vou avez appris à :

Compétence	Détail
Créer une vue de projection CDS	Pour les objets Travel et Booking
Ajouter des annotations UI	@UI.lineItem, @UI.selectionField, etc.
Activer la recherche libre	@Search.searchable, @Search.defaultSearchElement
Afficher les descriptions textuelles	@ObjectModel.text.element
Utiliser les aides à la valeur	@Consumption.valueHelpDefinition
Relier les vues entre elles	Via redirected to composition child et redirected to parent
Tester les projections	Activation et exécution de F8

3.5 Enrichissement de modèle de données CDS projeté avec des métadonnées UI (Unité 5)

Objectif

L'objectif de cette unité est de **compléter les vues de projection CDS** (ZC_) avec des **annotations UI**, sans toucher directement à leur définition, en utilisant les **extensions de métadonnées (Metadata Extensions - MDE)**. Cela prépare la génération d'une application SAP Fiori basée sur des éléments (Fiori Elements).

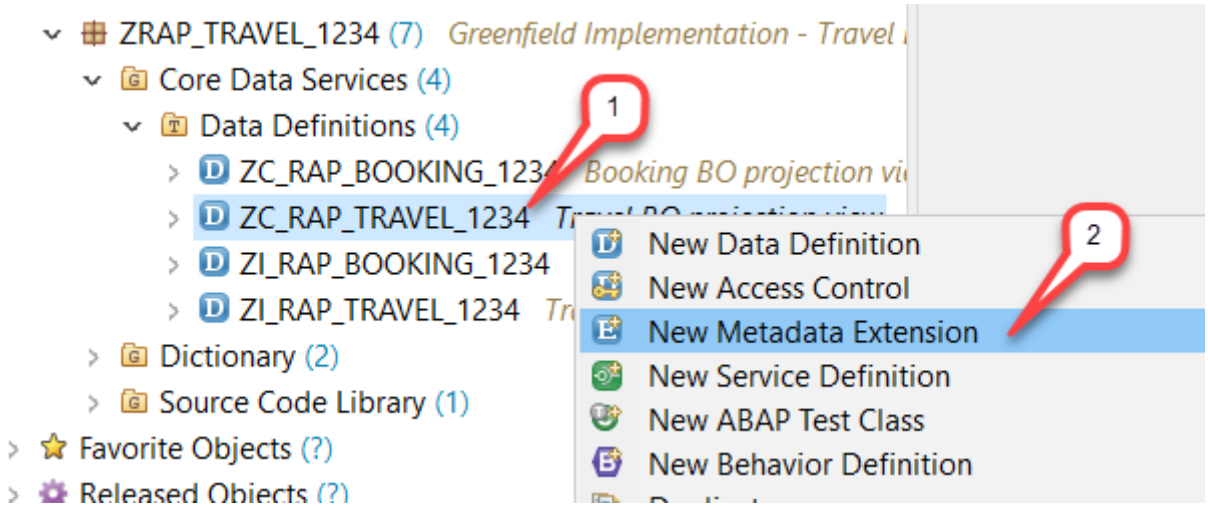
Étape 1 – Ajouter les annotations UI à ZC_RAP_Travel_####

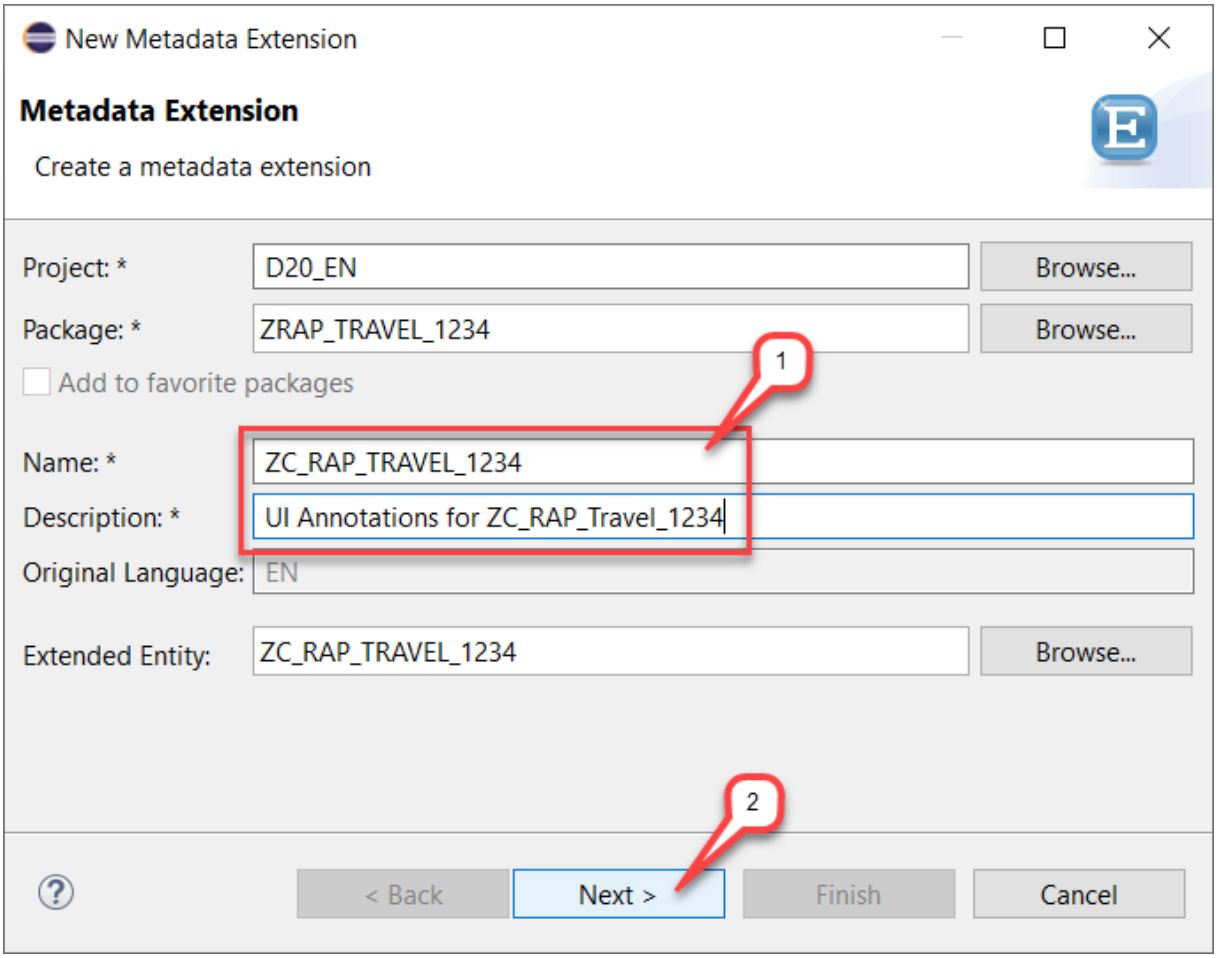
1. Clic droit sur la vue ZC_RAP_Travel_#### > **Nouvelle extension de métadonnées**
2. Choisissez un nom explicite : *ZC_RAP_Travel_#### (Annotations UI)*
3. Affectez un transport, cliquez sur **Suivant**, puis **Terminer**
4. Dans l'éditeur, remplacez le contenu avec l'extrait d'annotation UI fourni

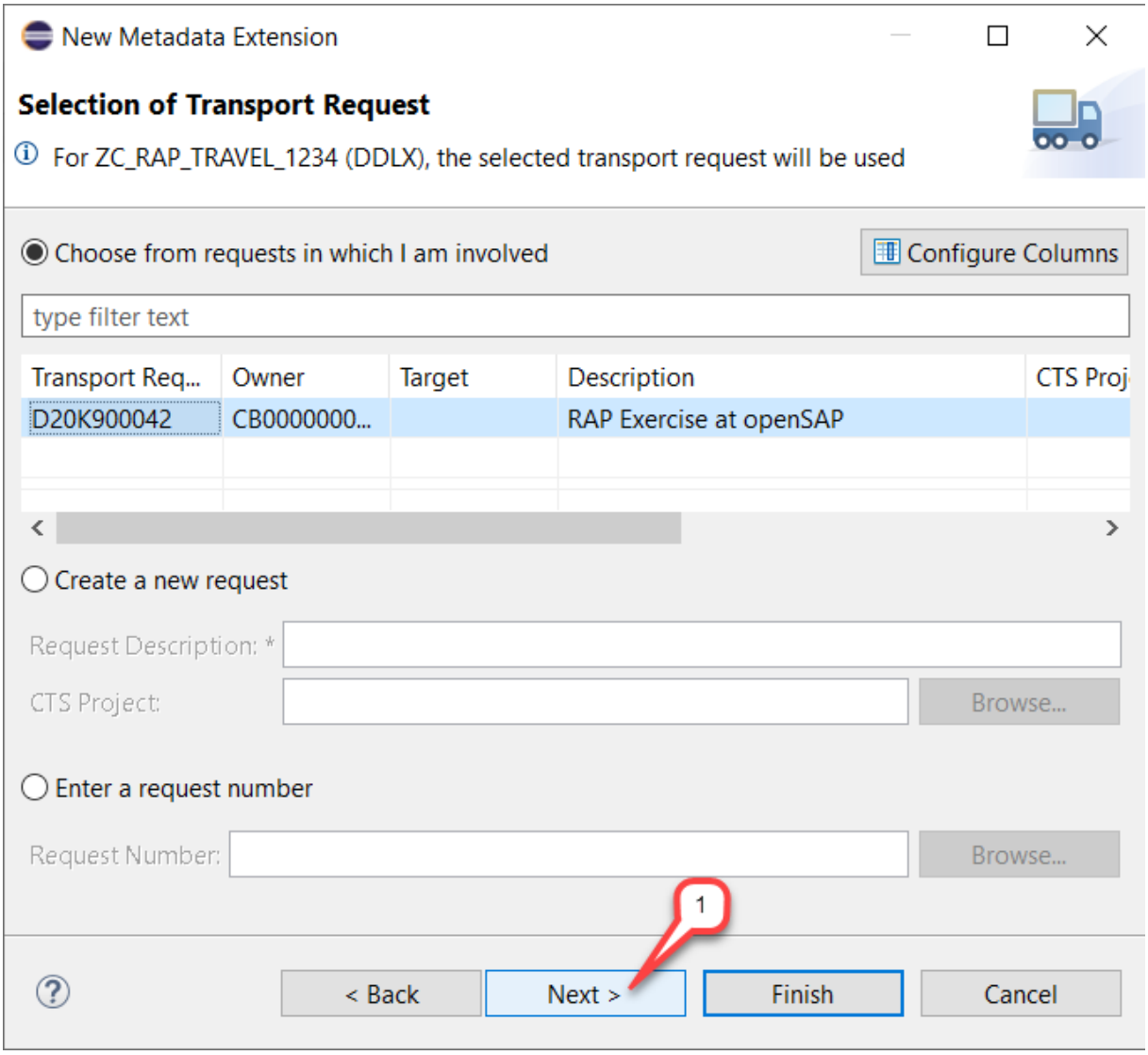
5. Supprimez de la liste les champs :
   - **AgencyName**, **CustomerName**, **CurrencyCode**
   - Les associations : **_Agency**, **_Customer**, **_Currency**, **_Booking**

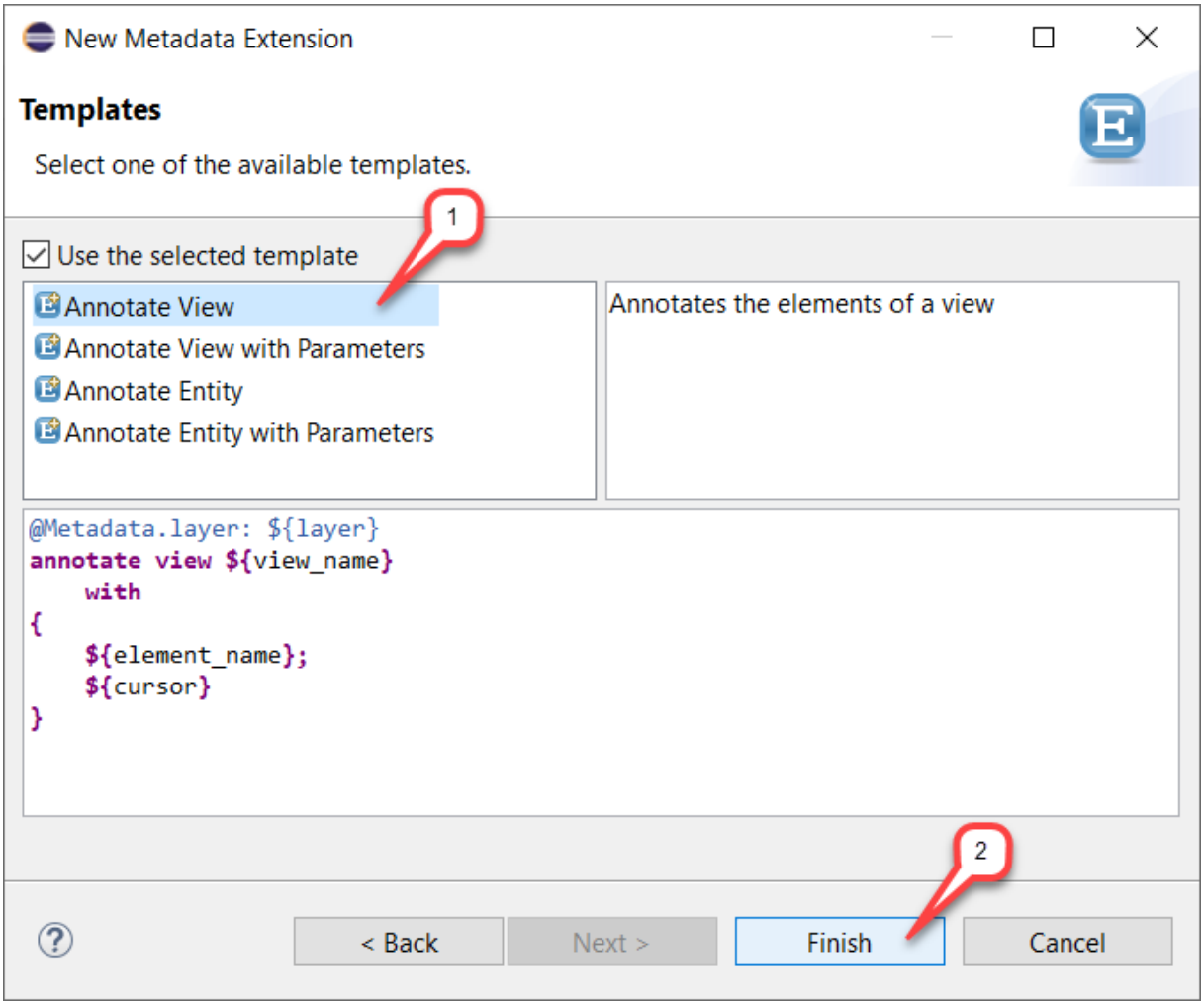
Pourquoi supprimer ces éléments ?

Ils sont déjà liés à leurs champs techniques (**AgencyID**, **CustomerID**, **CurrencyCode**) via les annotations **@ObjectModel.text.element** et **@Semantics.amount.currencyCode**, donc affichés automatiquement dans l'interface.




New Metadata Extension
Metadata Extension
Create a metadata extension
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
1
Name: *
ZC_RAP_TRAVEL_1234
Description: *
UI Annotations for ZC_RAP_Travel_1234
Original Language:
EN
Extended Entity:
ZC_RAP_TRAVEL_1234
Browse...
2
< Back
Next >
Finish
Cancel


New Metadata Extension
Selection of Transport Request
For ZC_RAP_TRAVEL_1234 (DDLX), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel



@Metadata.layer : #CORE

@UI: {

headerInfo : { typeName : 'Voyage',

typeNamePlural : « Voyages »,

titre : { type : #STANDARD, étiquette : 'Voyage', valeur : 'TravelID' } },

presentationVariant: [{ sortOrder: [{ by: 'TravelID', direction: #DESC }] }] }

annoter la vue ZC_RAP_Travel_#### avec

{

```
@UI.facet: [ { id: 'Voyage',
            objectif : #STANDARD,
            type : #IDENTIFICATION_REFERENCE,
            étiquette : « Voyage »,
            position: 10 },
           { id: 'Réservation',
            objectif : #STANDARD,
            type : #LINEITEM_REFERENCE,
            étiquette : « Réservation »,
            position: 20,
            targetElement : '_Réservation'} ]

@UI:{ identification: [{ position: 1, label: 'UUID de voyage' }] }
TravelUUID;

@UI: { lineItem: [ { position: 10 } ],
      identification : [ { position: 10 } ],
      champ de sélection : [ { position : 10 } ] }
Carte d'identité de voyage;

@UI: { lineItem: [ { position: 20 } ],
      identification : [ { position: 20 } ],
      champ de sélection : [ { position : 20 } ] }
ID d'agence;
```

```
@UI: { lineItem: [ { position: 30 } ],
      identification : [ { position: 30 } ],
      champ de sélection : [ { position : 30 } ] }
ID client ;

@UI: { lineItem: [ { position: 40 } ],
      identification : [ { position: 40 } ] }
Date de début;

@UI: { lineItem: [ { position: 50 } ],
      identification : [ { position: 50 } ] }
Date de fin;

@UI: { lineItem: [ { position: 60 } ],
      identification : [ { position: 60 } ] }
Frais de réservation ;

@UI: { lineItem: [ { position: 70 } ],
      identification : [ { position: 70 } ] }
Prix total;

@UI: { lineItem: [ { position: 80 } ],
      identification : [ { position: 80 } ] }
```

```
Description;


@UI: { lineItem: [ { position: 90 } ],
      identification : [ { position: 90 } ] }
Statut de voyage;


@UI.hidden : vrai
Dernière modification à ;


@UI.hidden : vrai
LocalLastChangedAt;
}
```

6. Enregistrez et activez les modifications.

Annotations utilisées

Annotation	Rôle
@UI.lineItem	Affichage dans la liste (table)
@UI.identification	Affichage sur la page de détail
@UI.selectionField	Ajoute le champ dans la barre de filtre
@UI.facet	Organisation des sections dans la page de détail
@UI.hidden: true	Masque un champ dans la UI Fiori

Annotation	Rôle
@Metadata.layer: #CORE	Spécifie que c'est une annotation fournie par le développeur de l'app (priorité faible)
@UI.headerInfo	Définit le titre principal et le tri de la liste

---

Résultat de l'enrichissement

- Liste triée par **TravelID** en ordre décroissant
- Deux sections dans la page d'objet :
  - Informations du voyage (**TravelID**, **BeginDate**, **EndDate**, etc.)
  - Liste liée des réservations (**_Booking**)
- Champs **LastChangedAt** et **LocalLastChangedAt** masqués via **@UI.hidden**

Étape 2 – Ajouter les annotations UI à ZC_RAP_Booking_####

1. Clic droit sur la vue ZC_RAP_Booking_#### > **Nouvelle extension de métadonnées**
2. Nom : *ZC_RAP_Booking_#### (Annotations UI)*
3. Affectez un transport, cliquez sur **Suivant**, puis **Terminer**
4. Dans l'éditeur, remplacez le contenu par les annotations fournies
5. Supprimez les champs :
   - **CustomerName**, **CarrierName**, **CurrencyCode**
   - Et les associations : **_Customer, _Carrier, _Connection, _Flight, _Travel**

```
@Metadata.layer : #CORE

@UI: {

 headerInfo : { typeName : 'Réservation',

            typeNamePlural : « Réservations »,
```

```
            titre : { type : #STANDARD, valeur : 'BookingID' } } }


annoter la vue ZC_RAP_Booking_####
  avec
{
 @UI.facet: [ { id: 'Réservation',
              objectif : #STANDARD,
              type : #IDENTIFICATION_REFERENCE,
              étiquette : « Réservation »,
              position: 10 } ]


 @UI : { identification : [ { position : 10, étiquette : 'UUID de réservation' } ] }
 RéservationUUID ;


 @UI.hidden : vrai
 TravelUUID;


 @UI: { lineItem: [ { position: 20 } ],
      identification : [ { position: 20 } ] }
 ID de réservation ;


 @UI: { lineItem: [ { position: 30 } ],
      identification : [ { position: 30 } ] }
 Date de réservation;


 @UI: { lineItem: [ { position: 40 } ],
```

```
    identification : [ { position: 40 } ] }
ID client ;

@UI: { lineItem: [ { position: 50 } ],
    identification : [ { position: 50 } ] }
ID du transporteur ;

@UI: { lineItem: [ { position: 60 } ],
    identification : [ { position: 60 } ] }
ID de connexion ;

@UI: { lineItem: [ { position: 70 } ],
    identification : [ { position: 70 } ] }
Date du vol;

@UI: { lineItem: [ { position: 80 } ],
    identification : [ { position: 80 } ] }
Prix du vol;

@UI.hidden : vrai
LocalLastChangedAt;
}
```

6. Enregistrez et activez les modifications.

Annotations UI utilisées

Annotation	Rôle
@UI.facet	Définit une section sur la page de détail
@UI.lineItem	Affichage dans la liste principale
@UI.identification	Affichage sur la page d'objet
@UI.hidden: true	Cache TravelUUID et LocalLastChangedAt
@UI.headerInfo	Définit le titre principal de la réservation (BookingID)

Résultat

- L'interface Fiori affichera :
  - Un tableau des réservations
  - Une page de détail par réservation
- Le champ **TravelUUID** (clé technique) et **LocalLastChangedAt** ne sont pas affichés à l'utilisateur

---

Résumé de l'unité

Élément appris	Description
Extension de métadonnées (MDE)	Ajout d'annotations UI sans modifier la projection principale
Personnalisation UI	Définir le tri, les colonnes, les filtres, les sections
Champs masqués	Pour éviter d'afficher des champs purement techniques
Lien vers vues enfants/parents	Navigation et affichage maîtrisés dans Fiori Elements
Usage SAP Fiori	Préparation back-end complète pour l'app Fiori de type **List Report / Object Page**

3.6 Développement et prévisualisation du service commercial (Unité 6)

Objectif

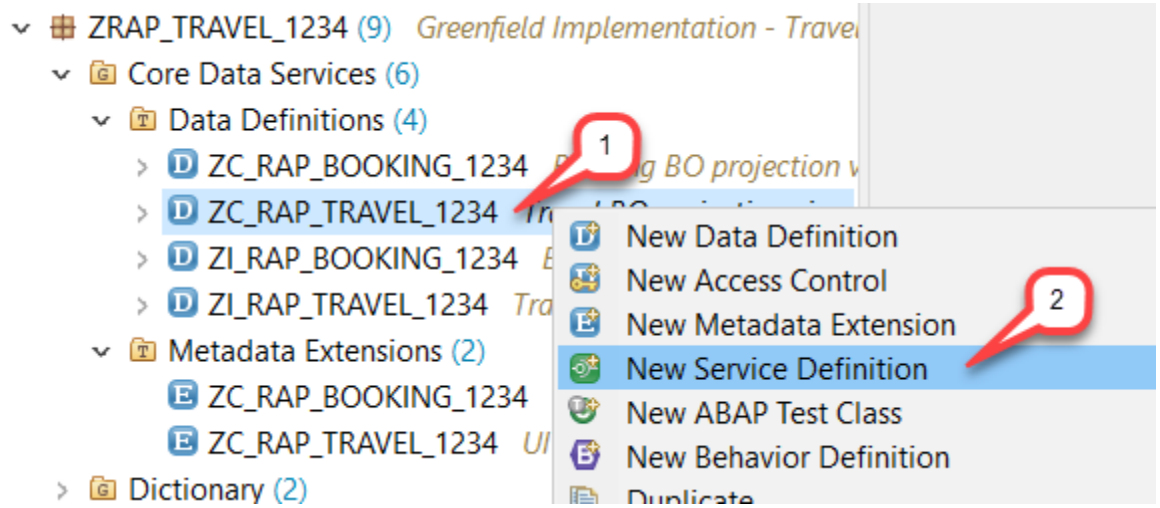
Dans cette unité, vous allez **exposer vos vues de projection CDS** (ZC_RAP_Travel_####, ZC_RAP_Booking_####) via un **service OData V2** à l'aide de :

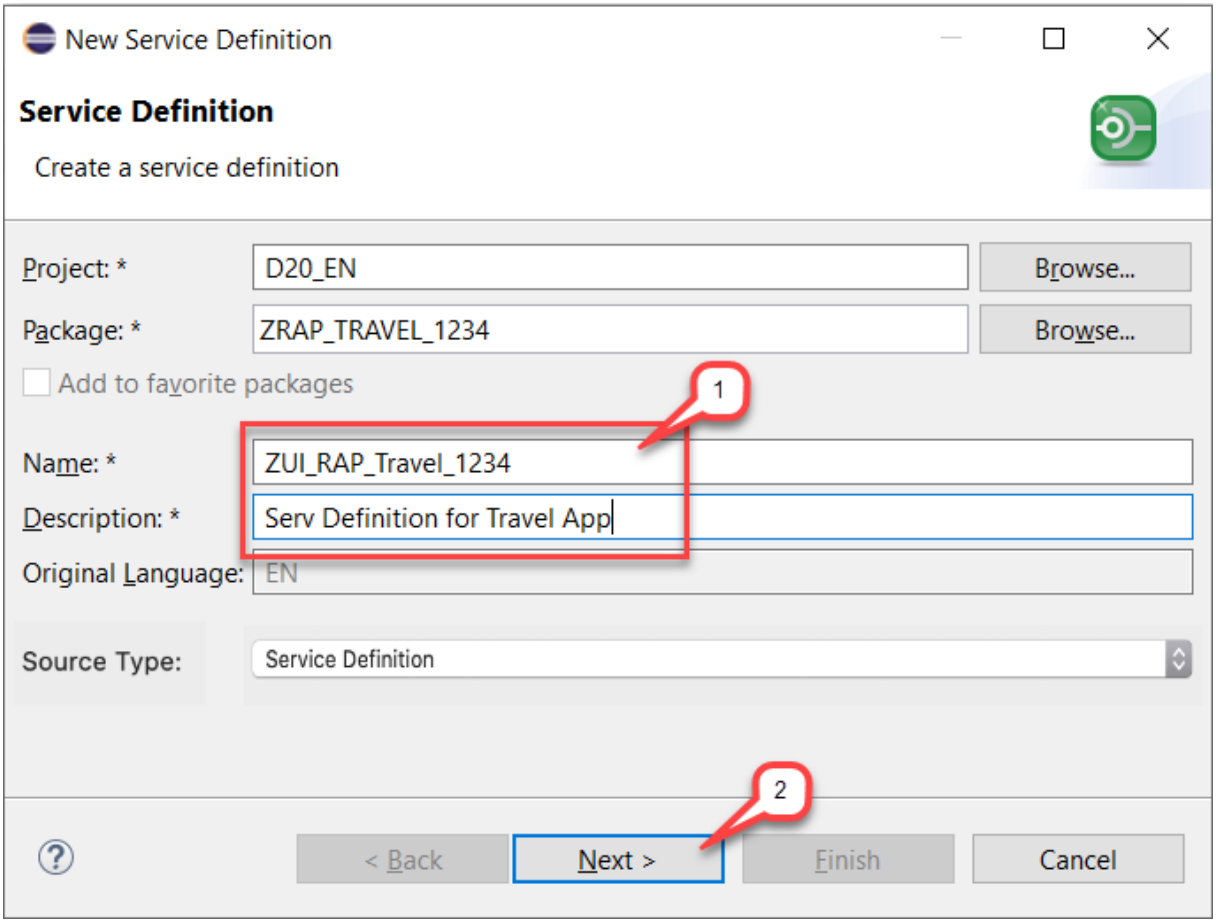
- **Une définition de service** (définit les entités exposées)
- **Une liaison de service** (connecte la définition au protocole OData)

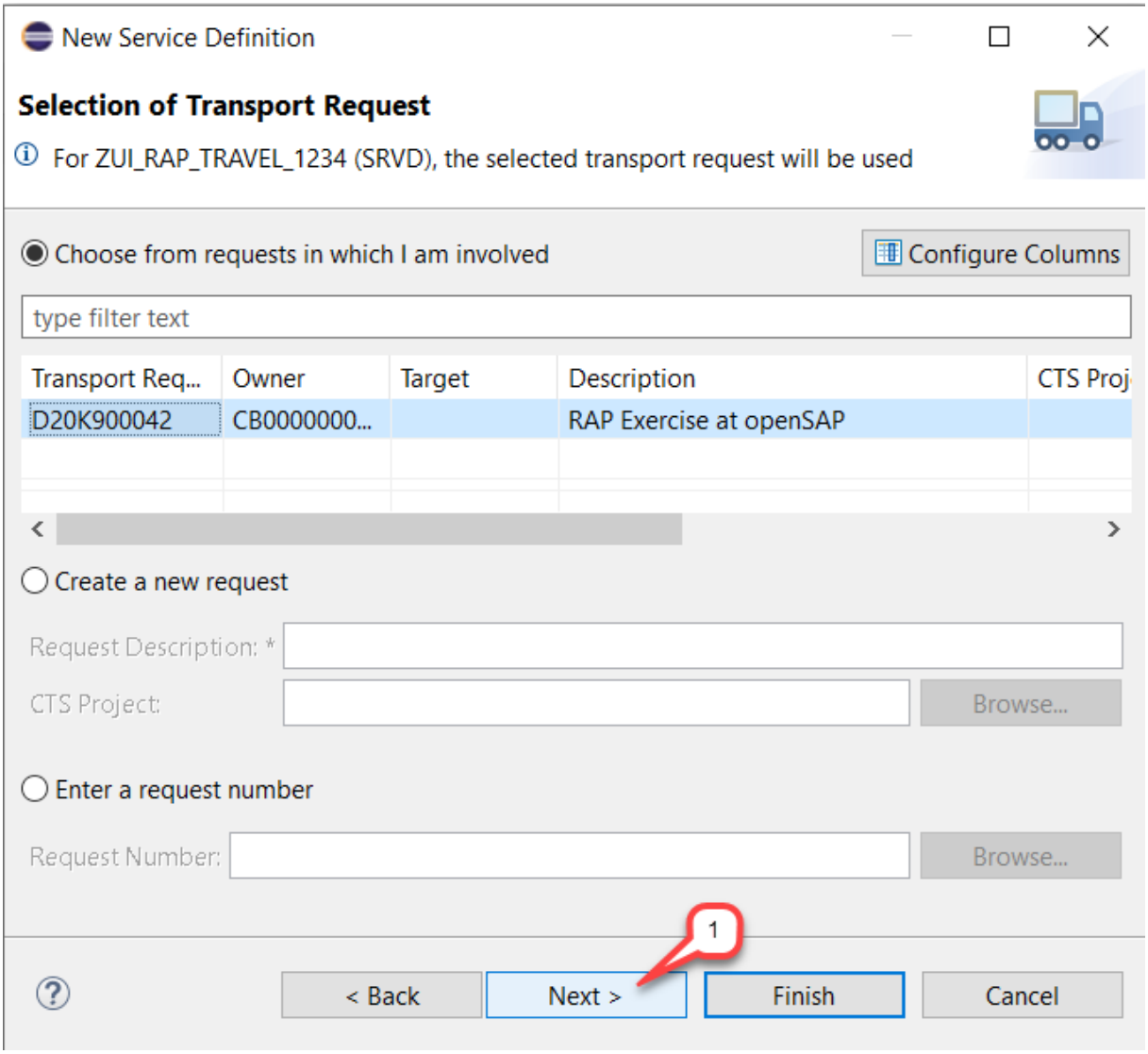
Vous pourrez ensuite **prévisualiser l'application SAP Fiori Elements** générée automatiquement pour tester votre modèle.

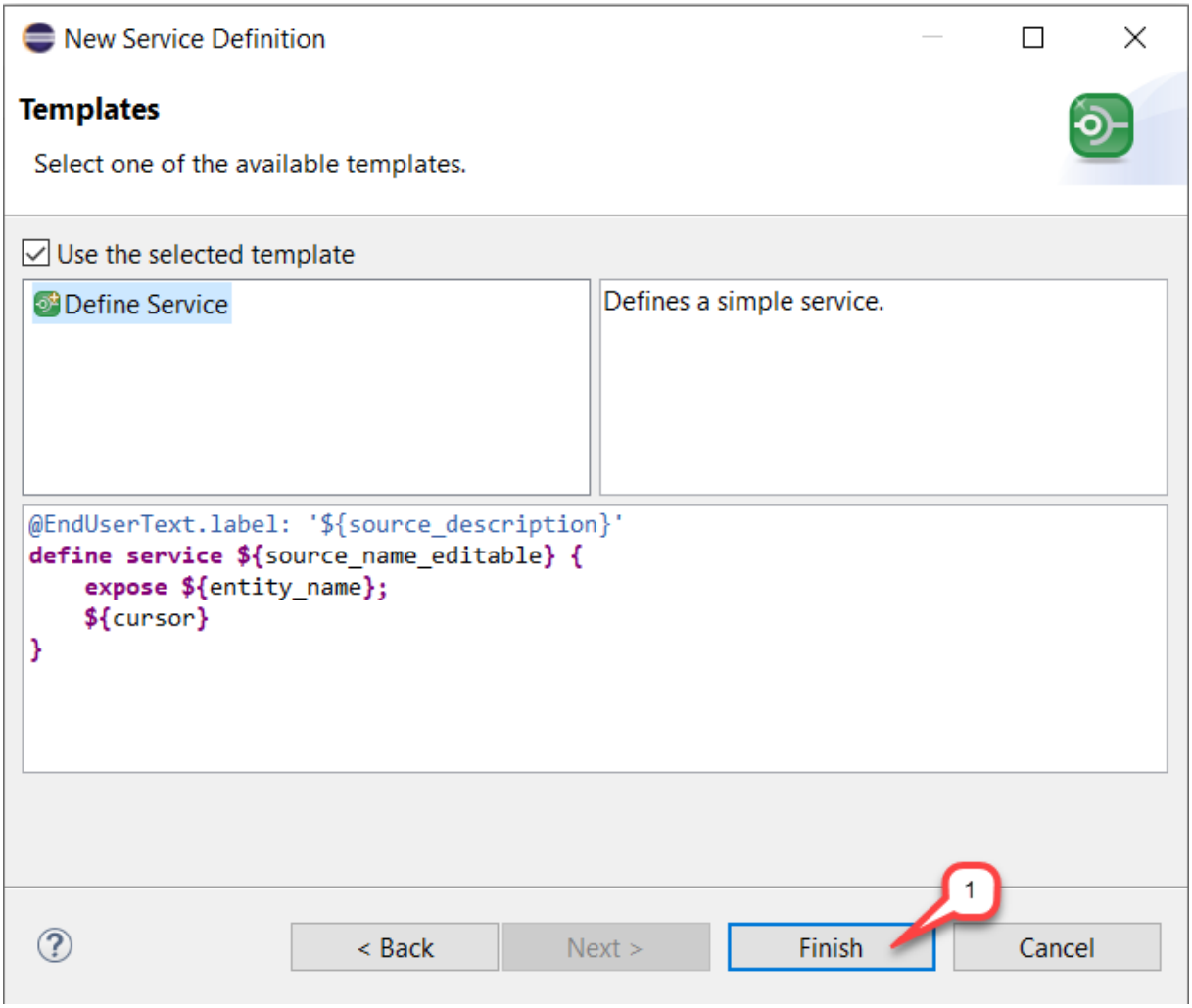
Étape 1 – Créer la définition de service ZUI_RAP_Travel_####

1. Clic droit sur ZC_RAP_Travel_#### > **Nouvelle définition de service**
2. Nom : ZUI_RAP_Travel_####
   Description : *Service Definition for Travel App*
3. Affectez une demande de transport
4. Choisissez le modèle **Define Service** > **Terminer**




New Service Definition
Service Definition
Create a service definition
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
1
Name: *
ZUI_RAP_Travel_1234
Description: *
Serv Definition for Travel App
Original Language:
EN
Source Type:
Service Definition
2
< Back
Next >
Finish
Cancel


New Service Definition
Selection of Transport Request
For ZUI_RAP_TRAVEL_1234 (SRVD), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel



Contenu à insérer

@EndUserText.label : « Définition de service pour l'application de voyage »

définir le service ZUI_RAP_TRAVEL_#### {

exposer ZC_RAP_Travel_#### comme Voyage ;

exposer ZC_RAP_Booking_#### comme Réservation ;

exposer /DMO/I_Agency comme Agence ;

exposer /DMO/I_Customer en tant que client ;

exposer /DMO/I_Flight comme Flight ;

exposer /DMO/I_Carrier comme Carrier ;

```
exposer /DMO/I_Connection comme connexion ;

exposer /DMO/I_Airport comme aéroport ;

exposer I_Currency comme devise ;

exposer I_Country comme pays ;

}
```

Remplace bien #### par votre suffixe personnel, et utilise Ctrl + F pour une recherche/remplacement rapide.

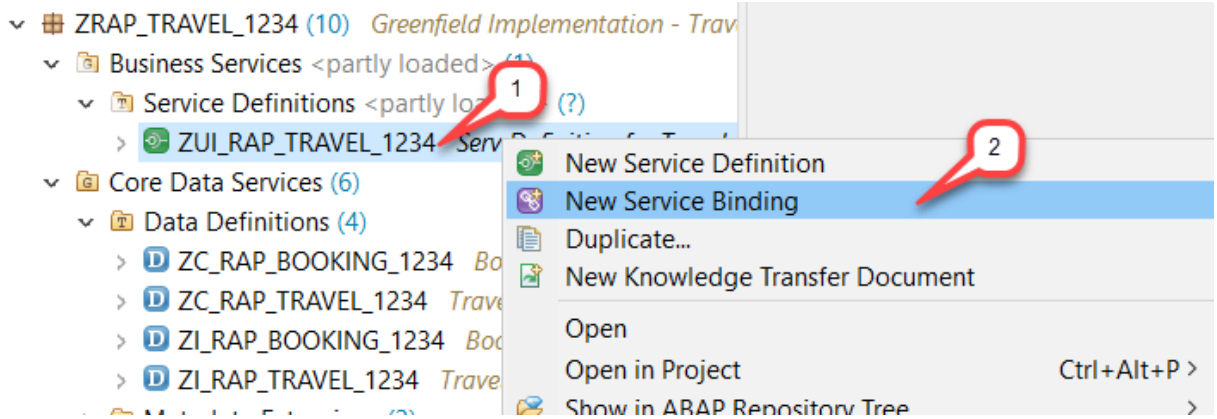
Ensuite :

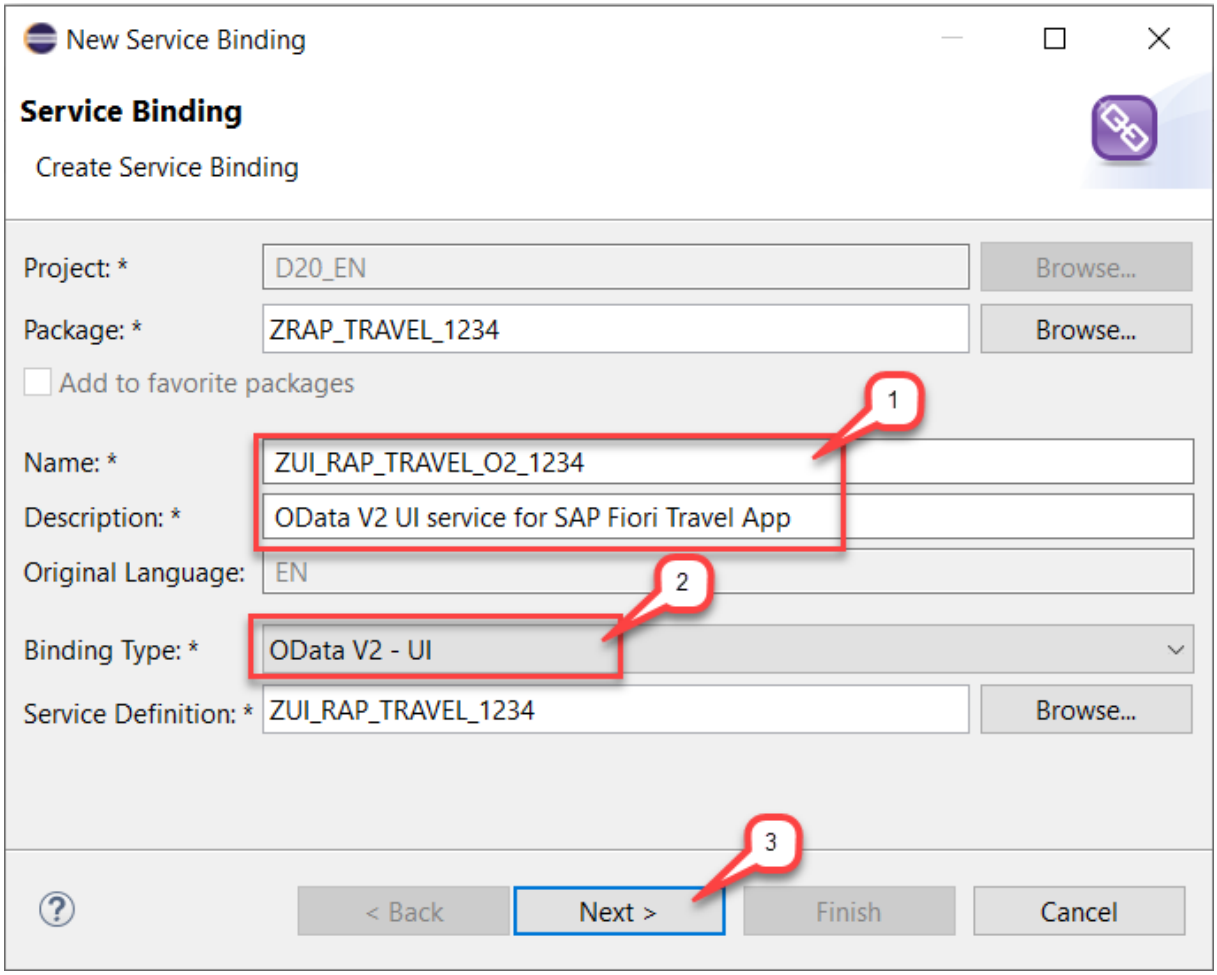
- Enregistrez et activez la définition du service. (**Ctrl + F3**)

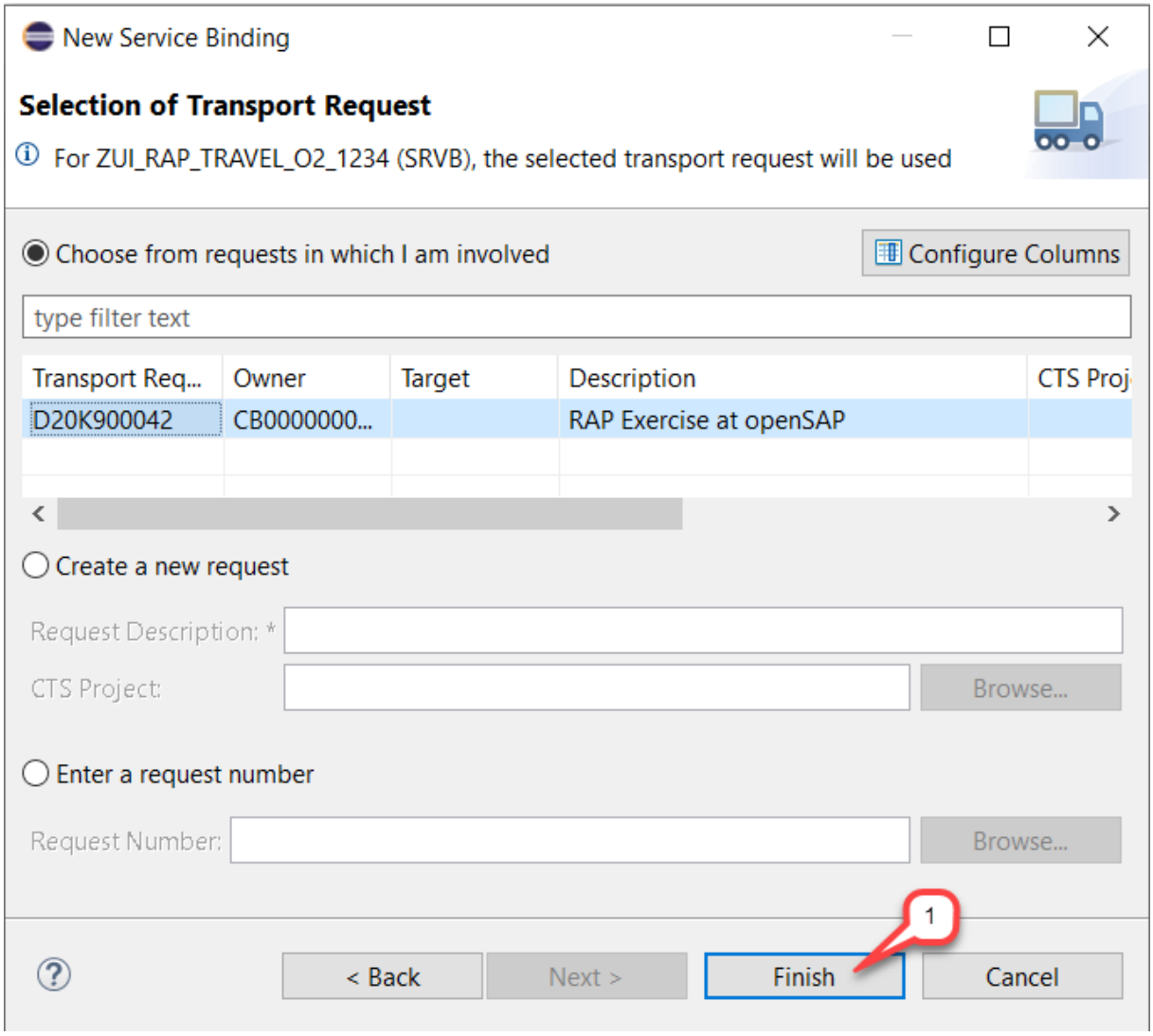
Étape 2 – Créer la liaison de service ZUI_RAP_TRAVEL_O2_####

Création

1. Clic droit sur ZUI_RAP_Travel_#### > **Nouvelle liaison de service**
2. Nom : ZUI_RAP_TRAVEL_O2_####
   Description : *OData V2 UI service for SAP Fiori Travel App*
3. Type de liaison : **OData V2 - UI**
4. Affectez une demande de transport > **Terminer**

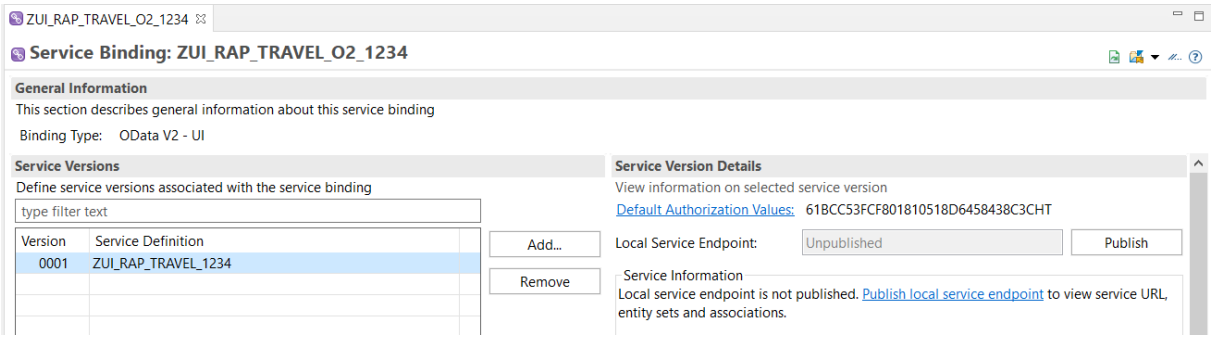



New Service Binding
Service Binding
Create Service Binding
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
1
Name: *
ZUI_RAP_TRAVEL_O2_1234
Description: *
OData V2 UI service for SAP Fiori Travel App
Original Language:
EN
2
Binding Type: *
OData V2 - UI
Service Definition: *
ZUI_RAP_TRAVEL_1234
Browse...
3
< Back
Next >
Finish
Cancel

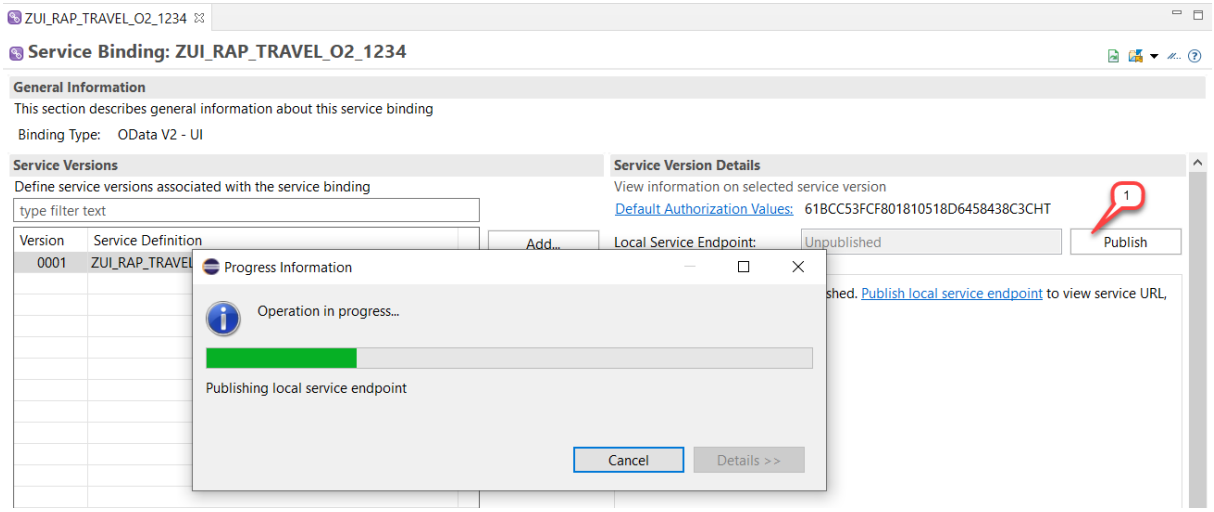


Activation

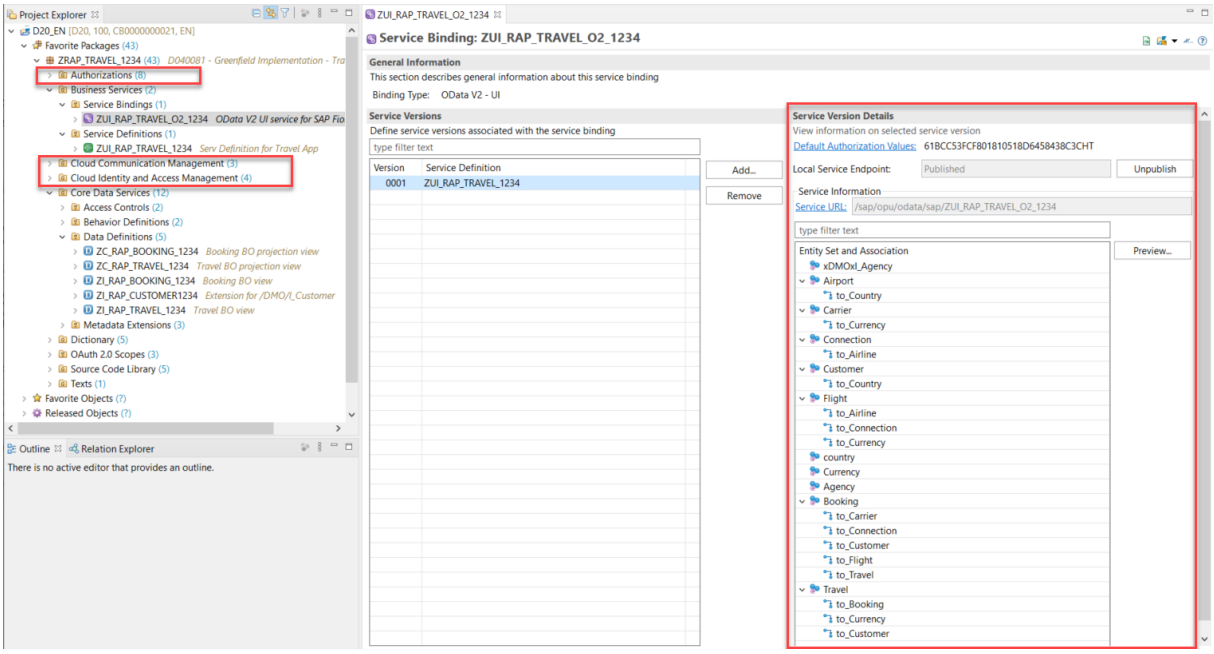
- Ouvrez la liaison dans l'éditeur



- Cliquez sur **Publish** (ou **Activate**) dans la section *Détails de la version du service*

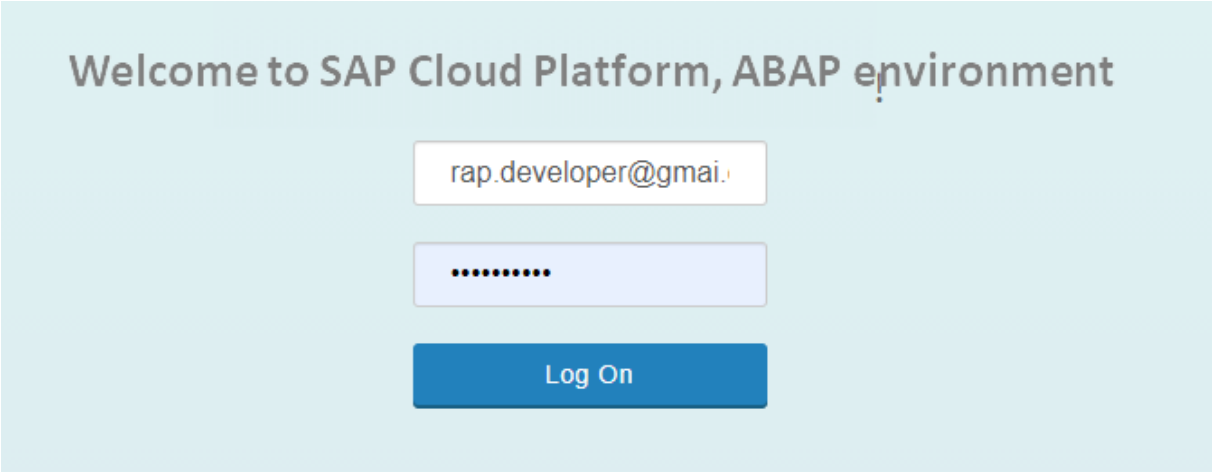


- Attendez la publication du **point de terminaison local OData**



Accès

- Cliquez sur l'**URL générée** pour voir le document de métadonnées OData ($metadata)
- Si besoin : saisir vos identifiants ABAP

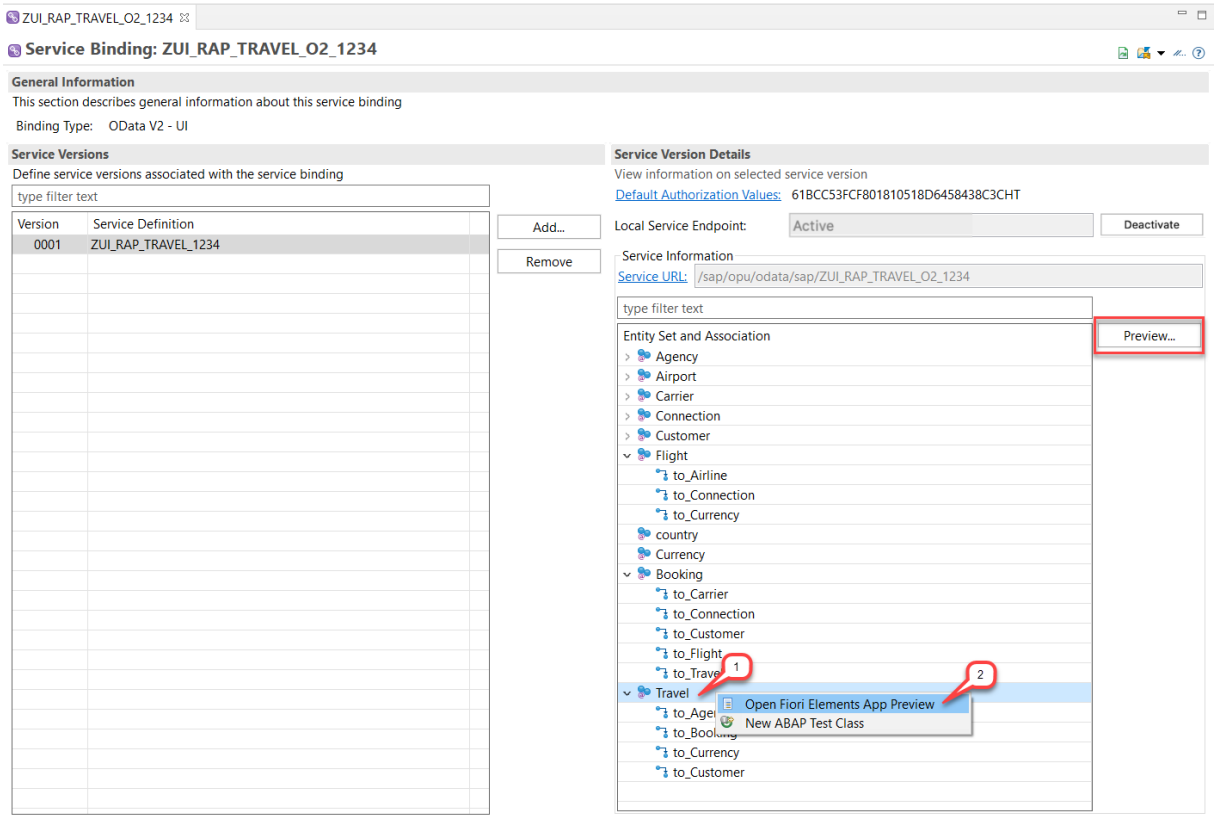



```
<app:service xml:lang="en" xml:base="https://
ZUI_RAP_TRAVEL_O2_1234/" xmlns:app="http://www.w3.org/2007/app"
xmlns:atom="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:sap="http://www.sap.com/Protocols/SAPData">
  <app:workspace>
    <atom:title type="text">Data</atom:title>
    <app:collection sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:searchable="true" sap:content-version="1"
    href="Agency">
      <atom:title type="text">Agency</atom:title>
      <sap:member-title>Agency View - CDS Data Model</sap:member-title>
      <atom:link href="Agency/OpenSearchDescription.xml" rel="search" type="application/opensearchdescription+xml" title="searchAgency"/>
    </app:collection>
    <app:collection sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:searchable="true" sap:content-version="1"
    href="Airport">
      <atom:title type="text">Airport</atom:title>
      <sap:member-title>Airport View - CDS Data Model</sap:member-title>
      <atom:link href="Airport/OpenSearchDescription.xml" rel="search" type="application/opensearchdescription+xml"
      title="searchAirport"/>
    </app:collection>
    <app:collection sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:searchable="true" sap:content-version="1"
    href="Carrier">
      <atom:title type="text">Carrier</atom:title>
      <sap:member-title>Carrier View - CDS Data Model</sap:member-title>
      <atom:link href="Carrier/OpenSearchDescription.xml" rel="search" type="application/opensearchdescription+xml"
      title="searchCarrier"/>
    </app:collection>
    <app:collection sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:searchable="true" sap:content-version="1"
    href="Connection">
```

Étape 3 – Prévisualiser l'application Fiori

Lancer la preview

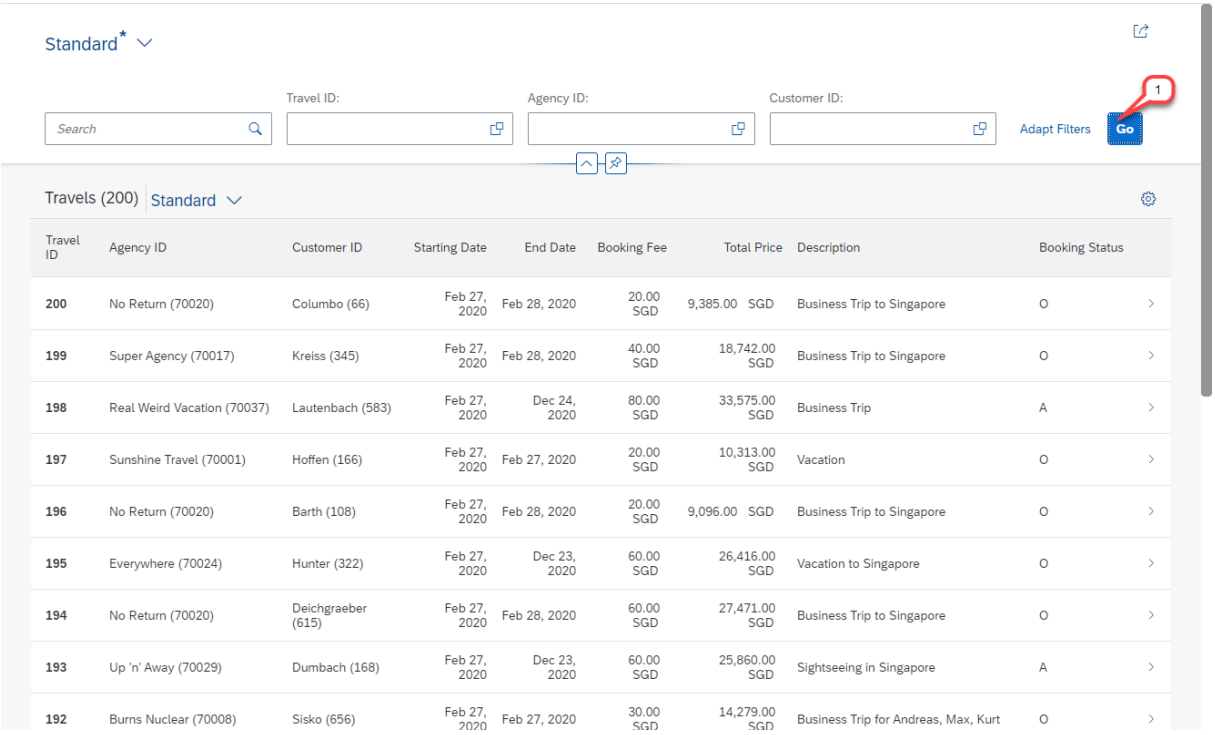
1. Dans l'éditeur de la liaison de service
2. Clic droit sur Travel > **Ouvrir l'aperçu de l'application des éléments Fiori**
   ou cliquez sur le bouton **Preview**



Tester l'application

Vous arrivez sur une **application Fiori générée automatiquement** :

- **Cliquez sur "Go"** pour charger les données
- Recherchez en texte libre (ex. : *Pan*)
- Utilisez les **filtres** (par AgencyID, CustomerID, etc.)
- Cliquez sur une ligne pour accéder à la **page de détails** du voyage
    - Deux sections : Travel (infos principales) + Booking (liste des réservations)
    - Cliquez sur une réservation pour voir ses propres détails



Travel ID	Agency ID	Customer ID	Starting Date	End Date	Booking Fee	Total Price	Description	Booking Status
200	No Return (70020)	Columbo (66)	Feb 27, 2020	Feb 28, 2020	20.00 SGD	9,385.00 SGD	Business Trip to Singapore	O
199	Super Agency (70017)	Kreiss (345)	Feb 27, 2020	Feb 28, 2020	40.00 SGD	18,742.00 SGD	Business Trip to Singapore	O
198	Real Weird Vacation (70037)	Lautenbach (583)	Feb 27, 2020	Dec 24, 2020	80.00 SGD	33,575.00 SGD	Business Trip	A
197	Sunshine Travel (70001)	Hoffen (166)	Feb 27, 2020	Feb 27, 2020	20.00 SGD	10,313.00 SGD	Vacation	O
196	No Return (70020)	Barth (108)	Feb 27, 2020	Feb 28, 2020	20.00 SGD	9,096.00 SGD	Business Trip to Singapore	O
195	Everywhere (70024)	Hunter (322)	Feb 27, 2020	Dec 23, 2020	60.00 SGD	26,416.00 SGD	Vacation to Singapore	O
194	No Return (70020)	Deichgraeber (615)	Feb 27, 2020	Feb 28, 2020	60.00 SGD	27,471.00 SGD	Business Trip to Singapore	O
193	Up 'n' Away (70029)	Dumbach (168)	Feb 27, 2020	Dec 23, 2020	60.00 SGD	25,860.00 SGD	Sightseeing in Singapore	A
192	Burns Nuclear (70008)	Sisko (656)	Feb 27, 2020	Feb 27, 2020	30.00 SGD	14,279.00 SGD	Business Trip for Andreas, Max, Kurt	O

Résumé de l'unité

Élément	Description
Définition de service	Spécifie les entités CDS projetées et leurs associations à exposer
Liaison de service	Rend le service disponible via OData V2
Publication	Active le point d'accès OData local
Aperçu Fiori Elements	Teste automatiquement l'UI générée à partir des métadonnées CDS
OData V2	Protocole utilisé pour exposer les données aux apps SAP Fiori

3.7 Mise en œuvre des autorisations de base dans RAP (Unité 7)

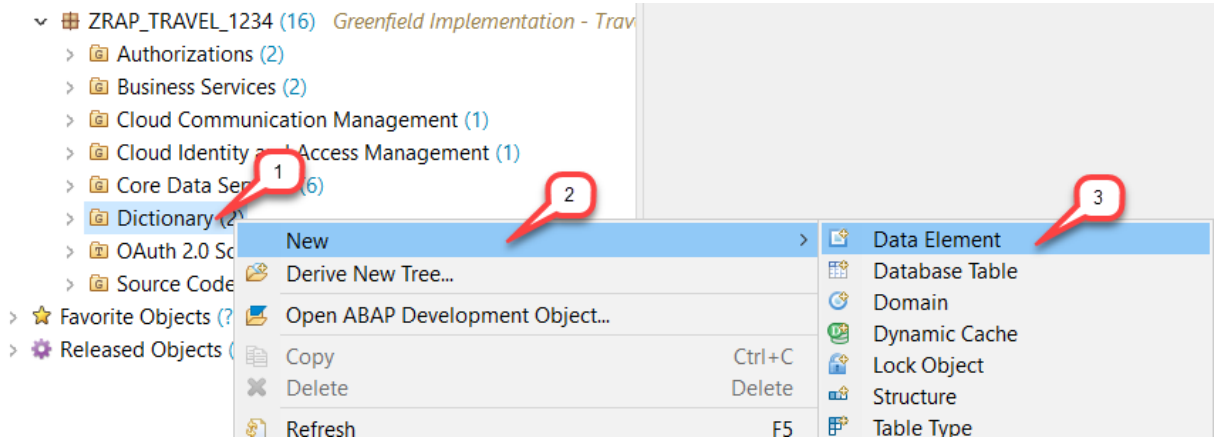
Objectif

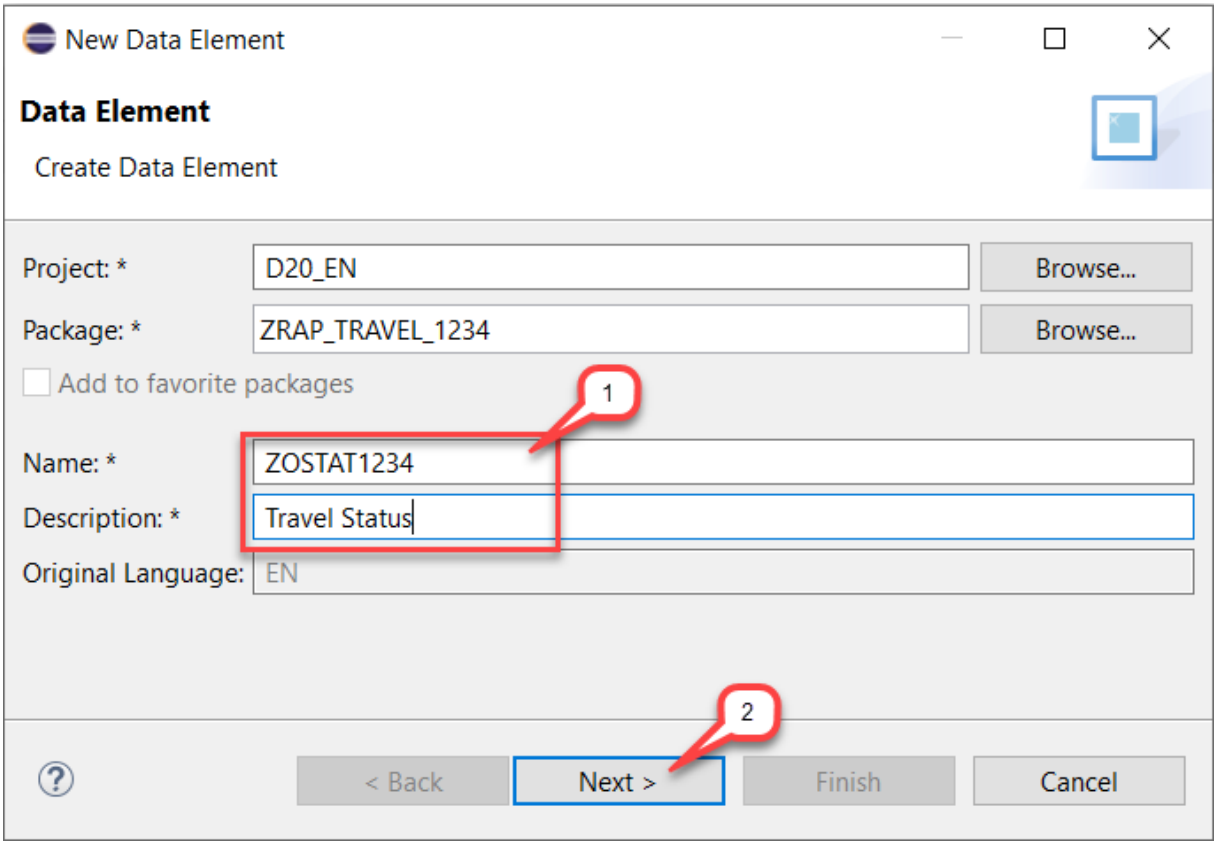
Dans cet exercice, vous allez **protéger l'accès aux données** de votre modèle CDS pour l'entité Travel en :

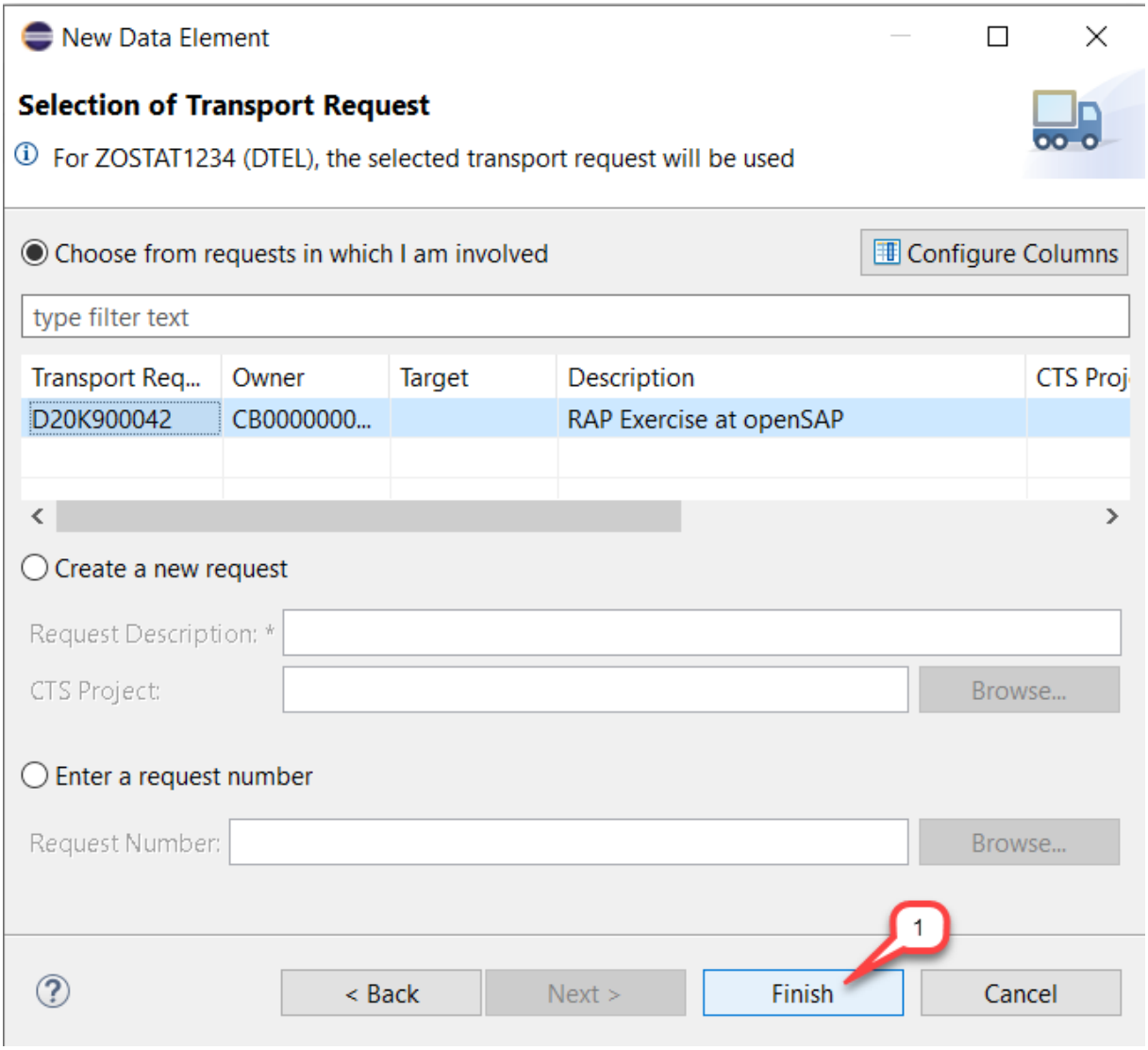
- Définissant un **élément de données**, un **champ d'autorisation** et un **objet d'autorisation** pour le champ TravelStatus.
- Créant des **règles d'accès (CDS Access Controls)** avec :
  - Une **condition littérale** sur CurrencyCode (filtrée sur 'EUR')
  - Une **condition PFCG** (vérifie les droits sur le TravelStatus de l'utilisateur)

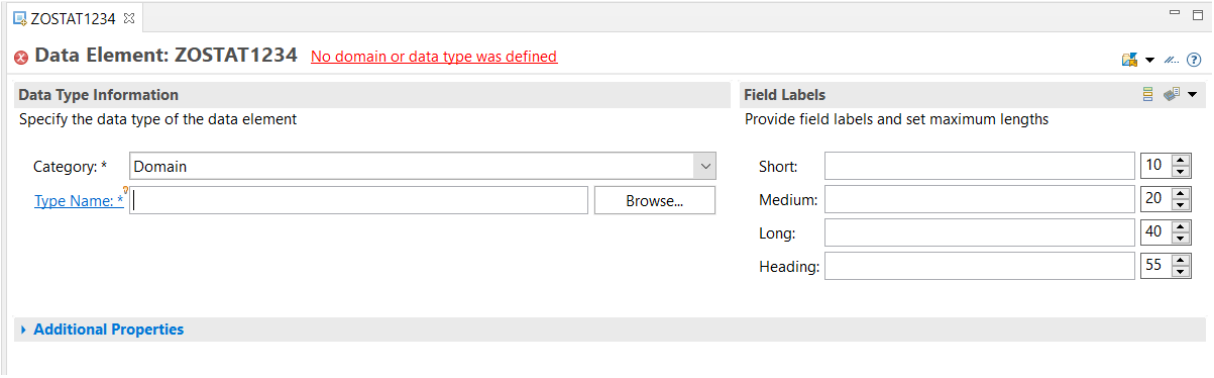
Étape 1 – Créer l'élément de données ZOSTAT####

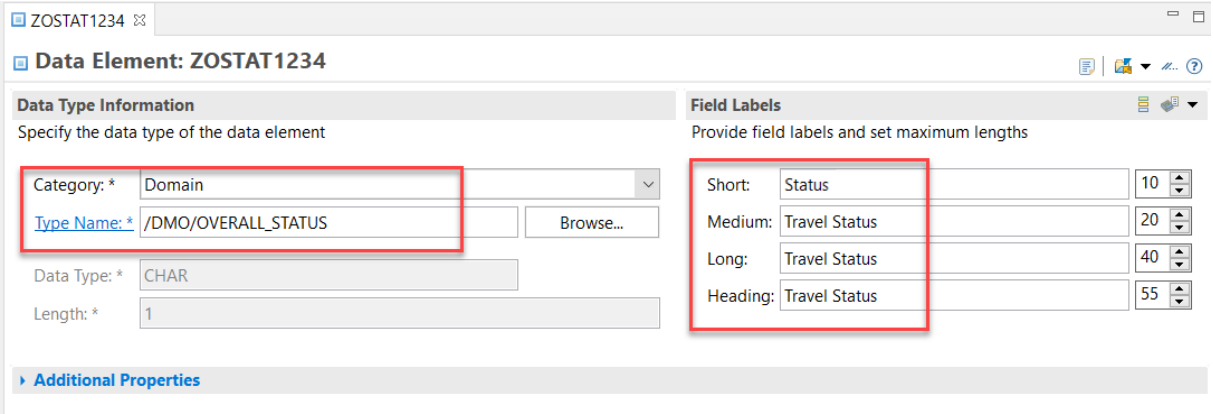
- Clic droit sur Dictionnaire > New > Data Element
- Nom : ZOSTAT####
- Description : *Travel Status*
- Domaine : /DMO/OVERALL_STATUS
- Étiquettes :
  - Courte : Status
  - Moyenne : Travel Status
  - Longue : Travel Status


ZRAP_TRAVEL_1234 (16) Greenfield Implementation - Trav
Authorizations (2)
Business Services (2)
Cloud Communication Management (1)
Cloud Identity and Access Management (1)
Core Data Ser (6)
Dictionary (2)
OAuth 2.0 Sc
Source Code
Favorite Objects (?
Released Objects (
1
2
3
New
Derive New Tree...
Open ABAP Development Object...
Copy Ctrl+C
Delete Delete
Refresh F5
Data Element
Database Table
Domain
Dynamic Cache
Lock Object
Structure
Table Type


New Data Element
Data Element
Create Data Element
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
1
Name: *
ZOSTAT1234
Description: *
Travel Status
Original Language:
EN
2
< Back
Next >
Finish
Cancel


New Data Element
Selection of Transport Request
For ZOSTAT1234 (DTEL), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel


ZOSTAT1234
Data Element: ZOSTAT1234
No domain or data type was defined
Data Type Information
Specify the data type of the data element
Category: *
Domain
Type Name: *
Browse...
Field Labels
Provide field labels and set maximum lengths
Short:
10
Medium:
20
Long:
40
Heading:
55
Additional Properties

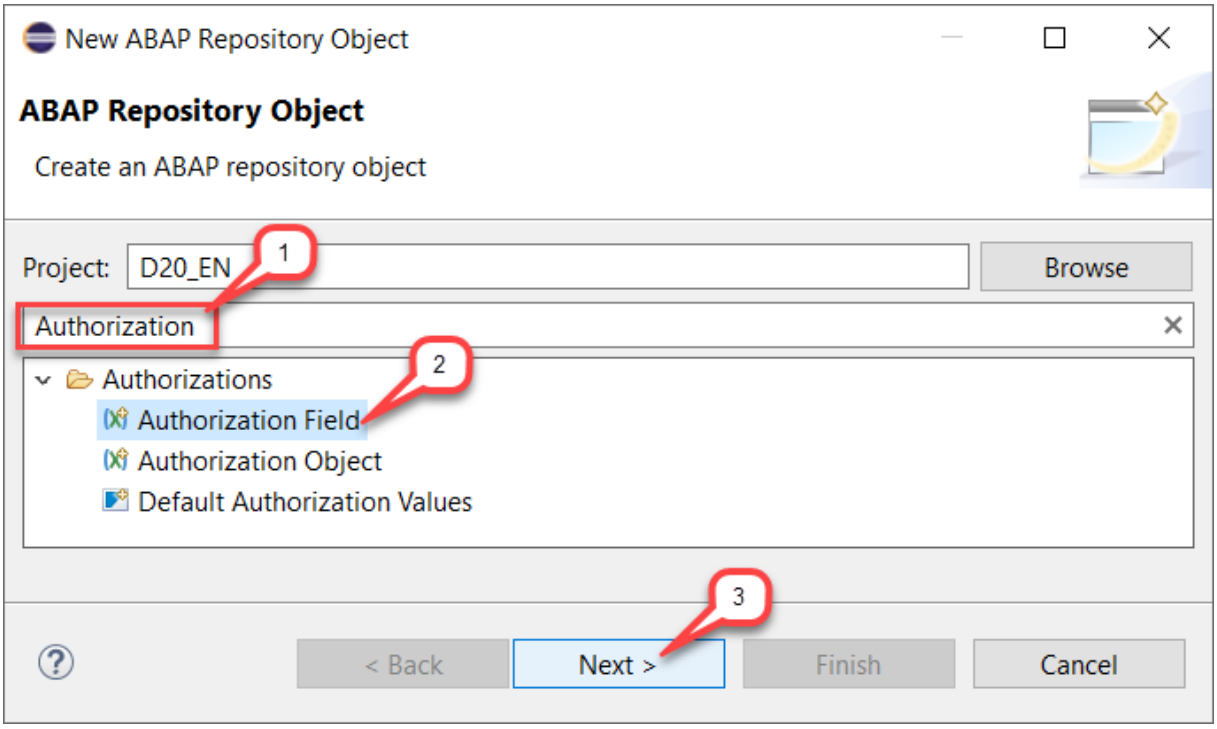


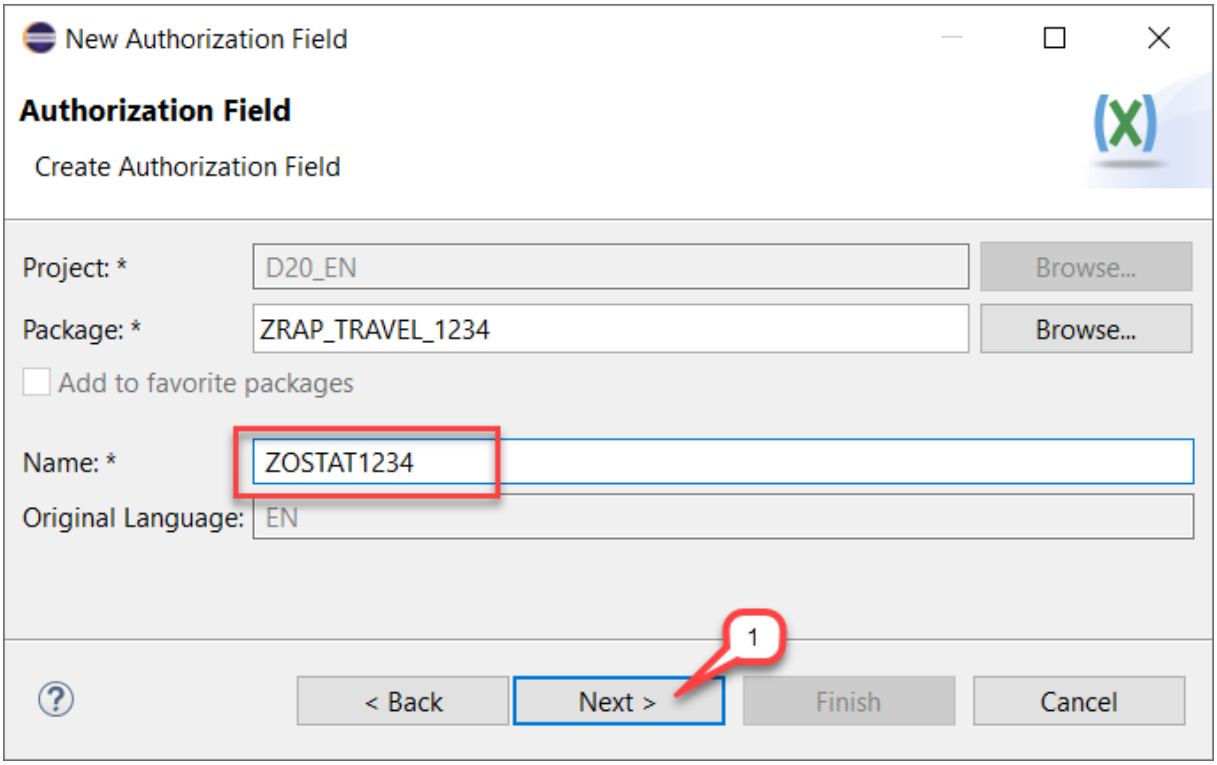
Enregistrez et activez le nouvel élément de données.

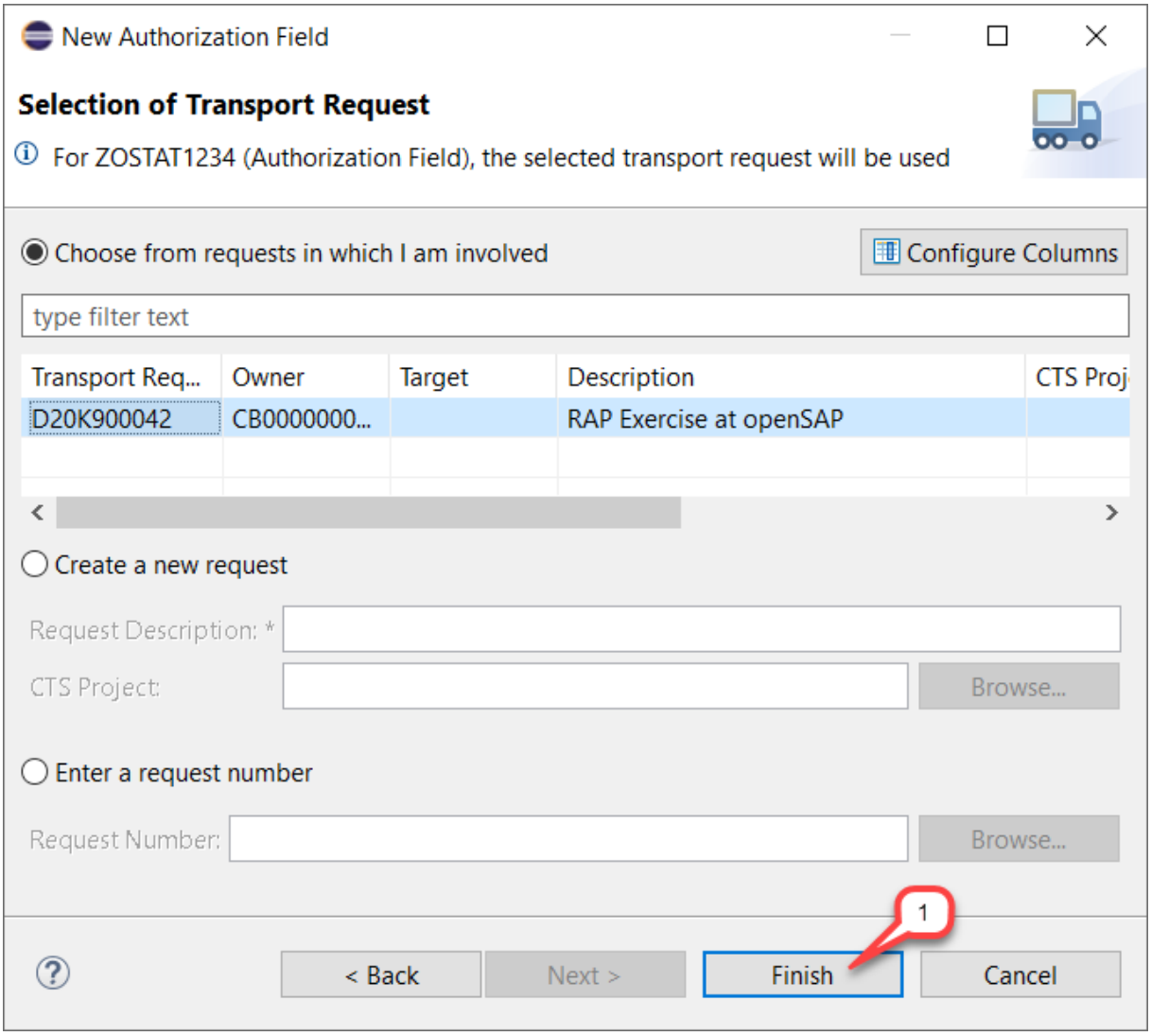
Étape 2 – Créer le champ d'autorisation ZOSTAT####

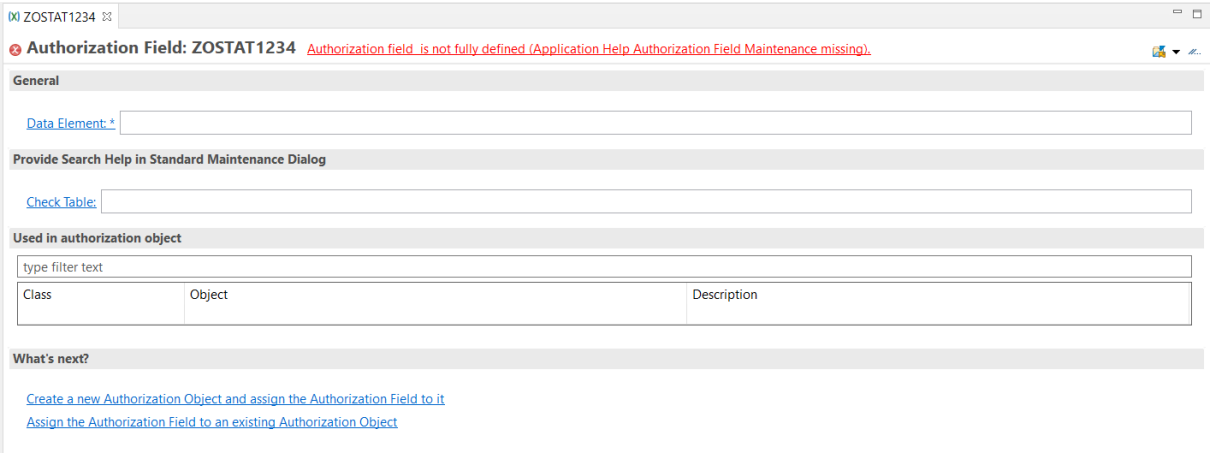
Création

- Clic droit sur le package > New > Other ABAP Repository Object
- Filtrez : Authorization Field
- Nom : ZOSTAT####
- Élément de données associé : ZOSTAT####

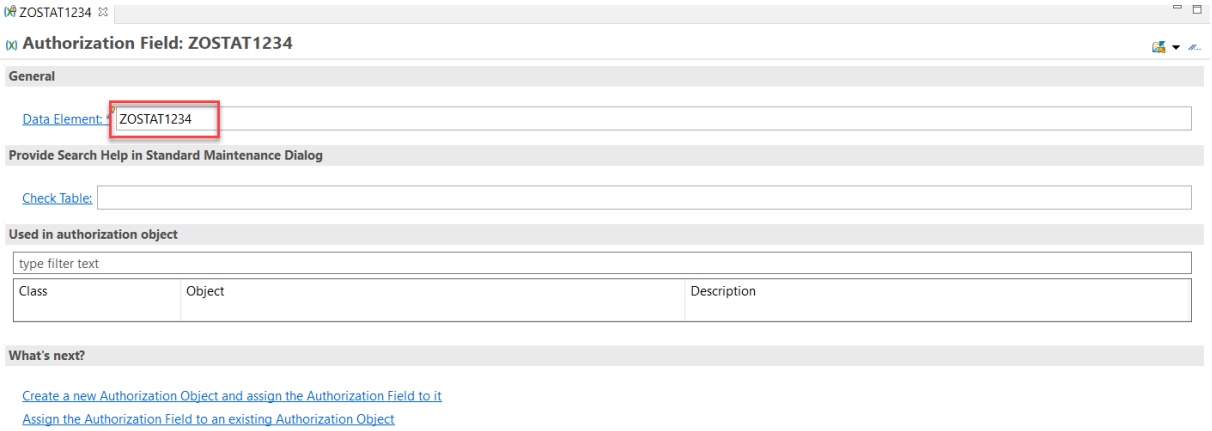

New ABAP Repository Object
ABAP Repository Object
Create an ABAP repository object
Project:
D20_EN
Browse
Authorization
Authorizations
Authorization Field
Authorization Object
Default Authorization Values
1
2
3
< Back
Next >
Finish
Cancel


New Authorization Field
Authorization Field
Create Authorization Field
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
Name: *
ZOSTAT1234
Original Language:
EN
1
< Back
Next >
Finish
Cancel


New Authorization Field
Selection of Transport Request
For ZOSTAT1234 (Authorization Field), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel


ZOSTAT1234
Authorization Field: ZOSTAT1234
Authorization field is not fully defined (Application Help Authorization Field Maintenance missing).
General
Data Element: *
Provide Search Help in Standard Maintenance Dialog
Check Table:
Used in authorization object
type filter text
Class
Object
Description
What's next?
Create a new Authorization Object and assign the Authorization Field to it
Assign the Authorization Field to an existing Authorization Object

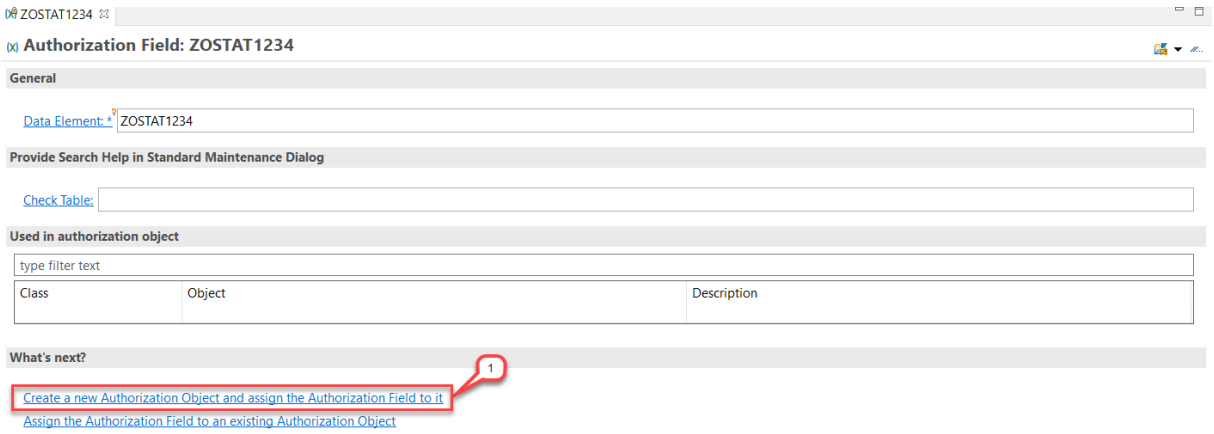
- Enregistrez le nouveau champ d'autorisation. Il sera automatiquement activé.
  La création du champ d'autorisation est terminée.

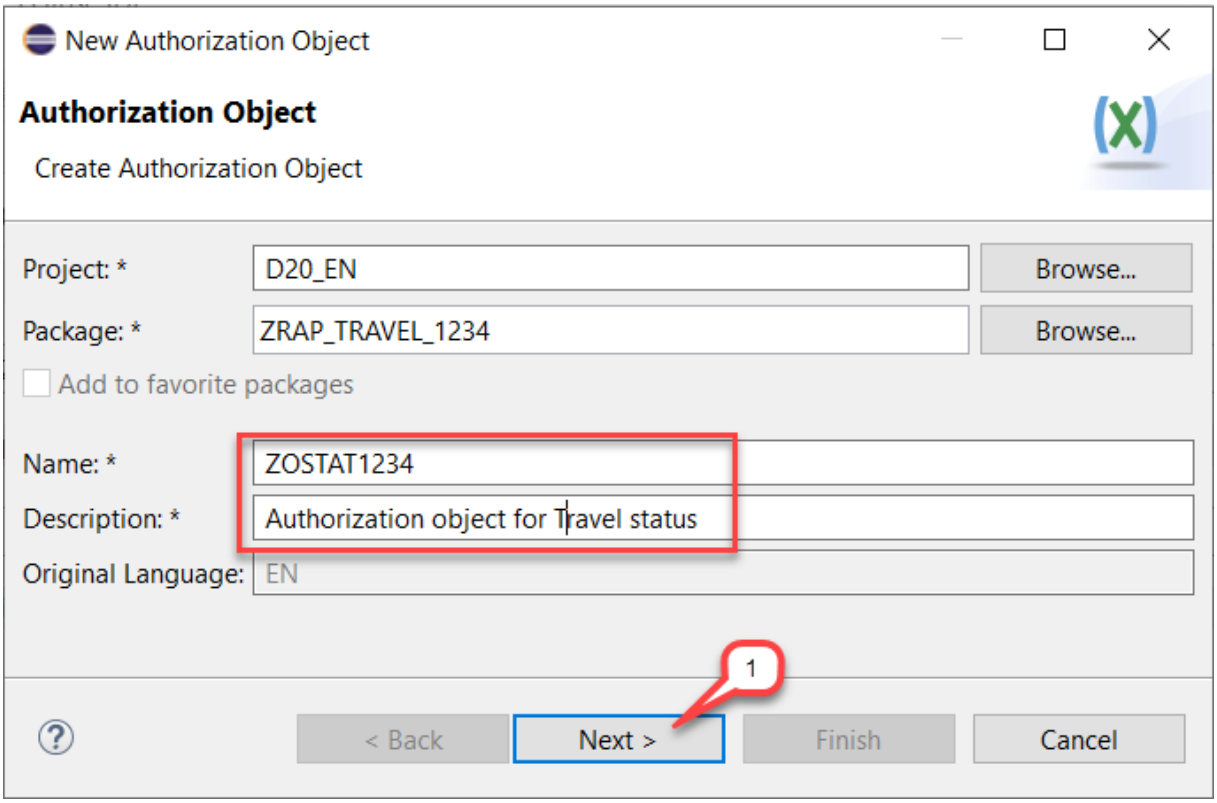


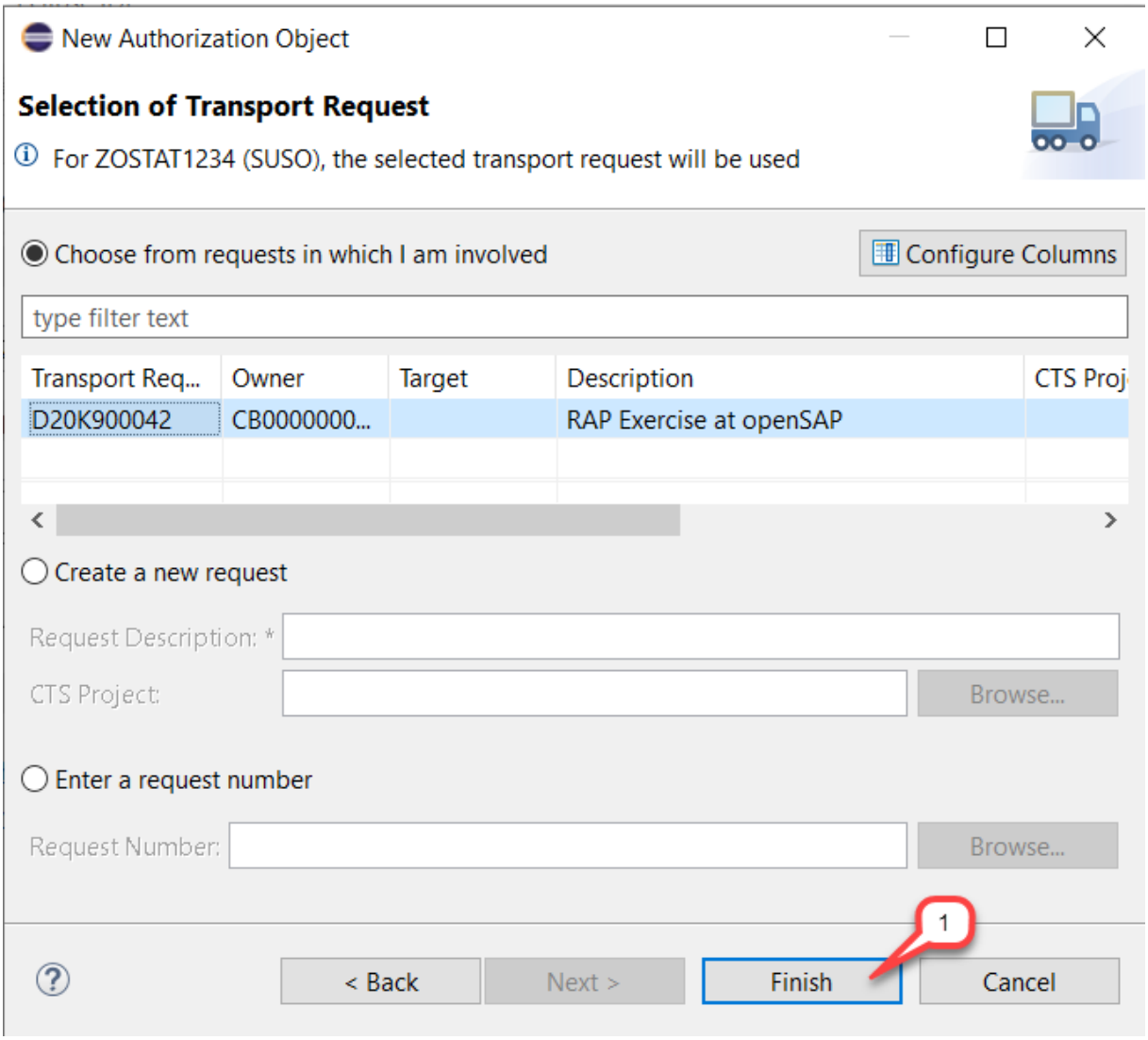
Étape 3 – Créer l'objet d'autorisation ZOSTAT####

Création

- Depuis le champ d'autorisation > **Créer un objet d'autorisation associé**
- Nom : ZOSTAT####
- Description: *Authorization object for Travel Status*

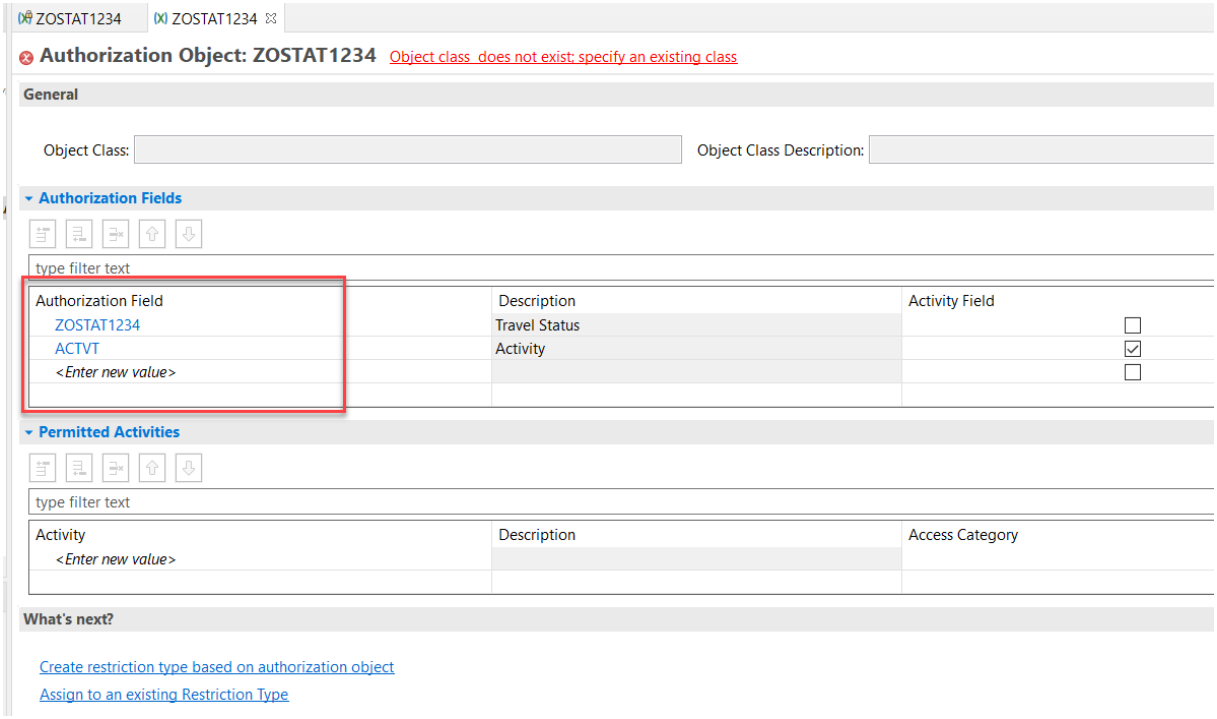



New Authorization Object
Authorization Object
Create Authorization Object
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
Name: *
ZOSTAT1234
Description: *
Authorization object for Travel status
Original Language:
EN
1
< Back
Next >
Finish
Cancel



Configuration

- Champs :
  - ACTVT (activités standards)
  - ZOSTAT#### (champ d'autorisation créé)



- Activités autorisées :

01 – Créer

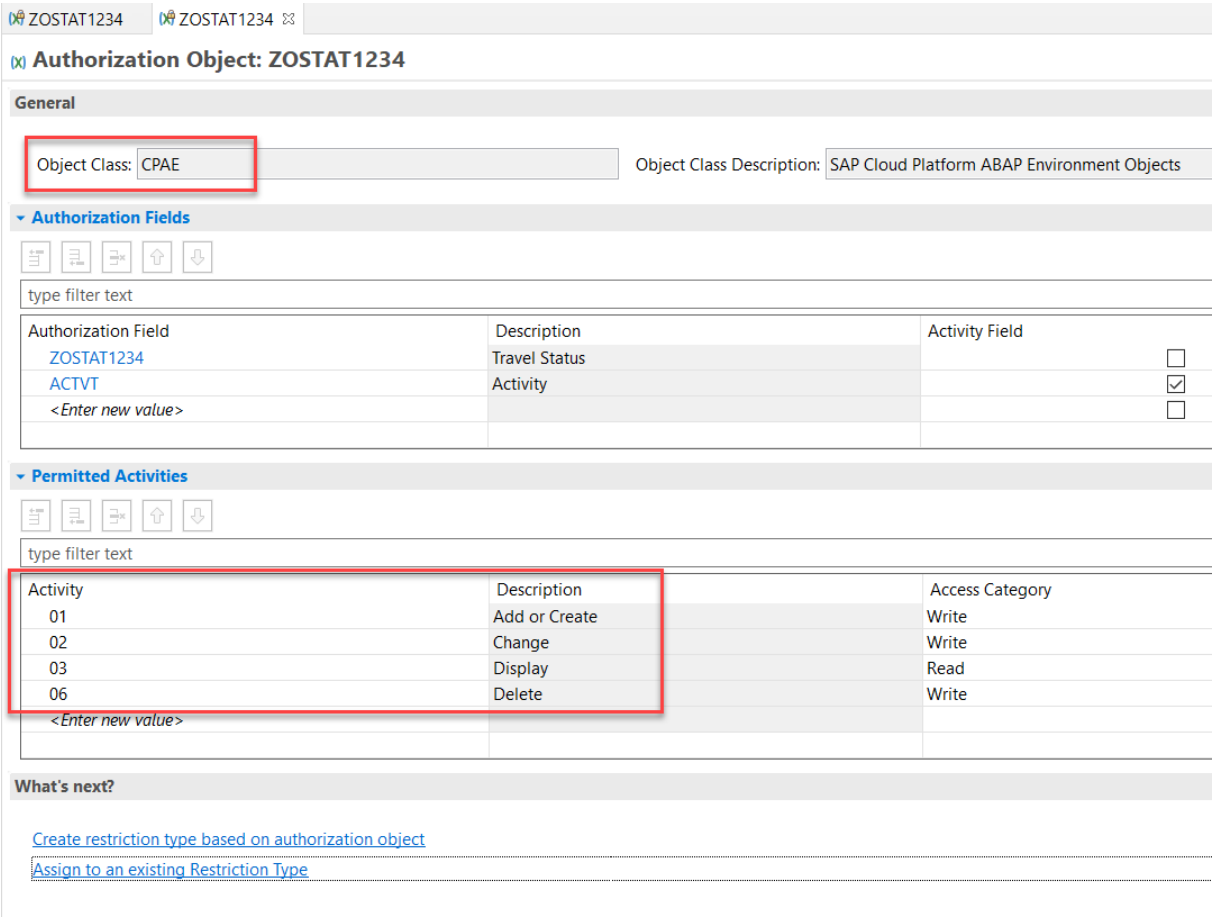
02 – Modifier

03 – Lire

06 – Supprimer

- Enregistrez les modifications pour activer l'objet d'autorisation. La classe d'objet sera automatiquement maintenue.

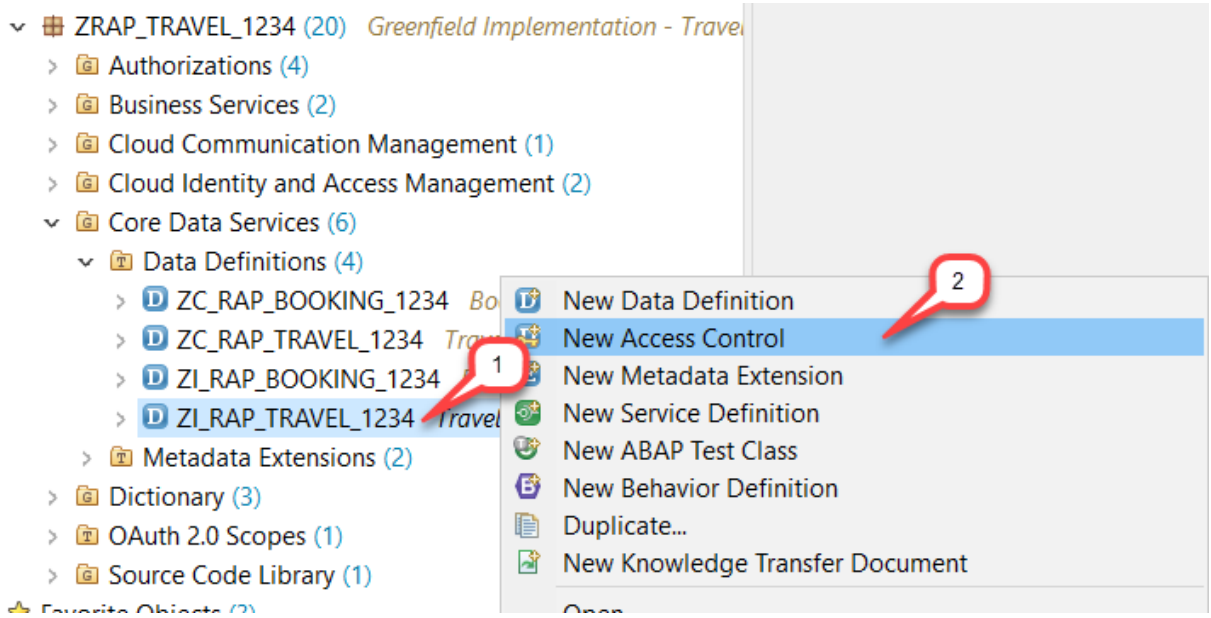
Vous avez maintenant terminé la création de l'objet d'autorisation.

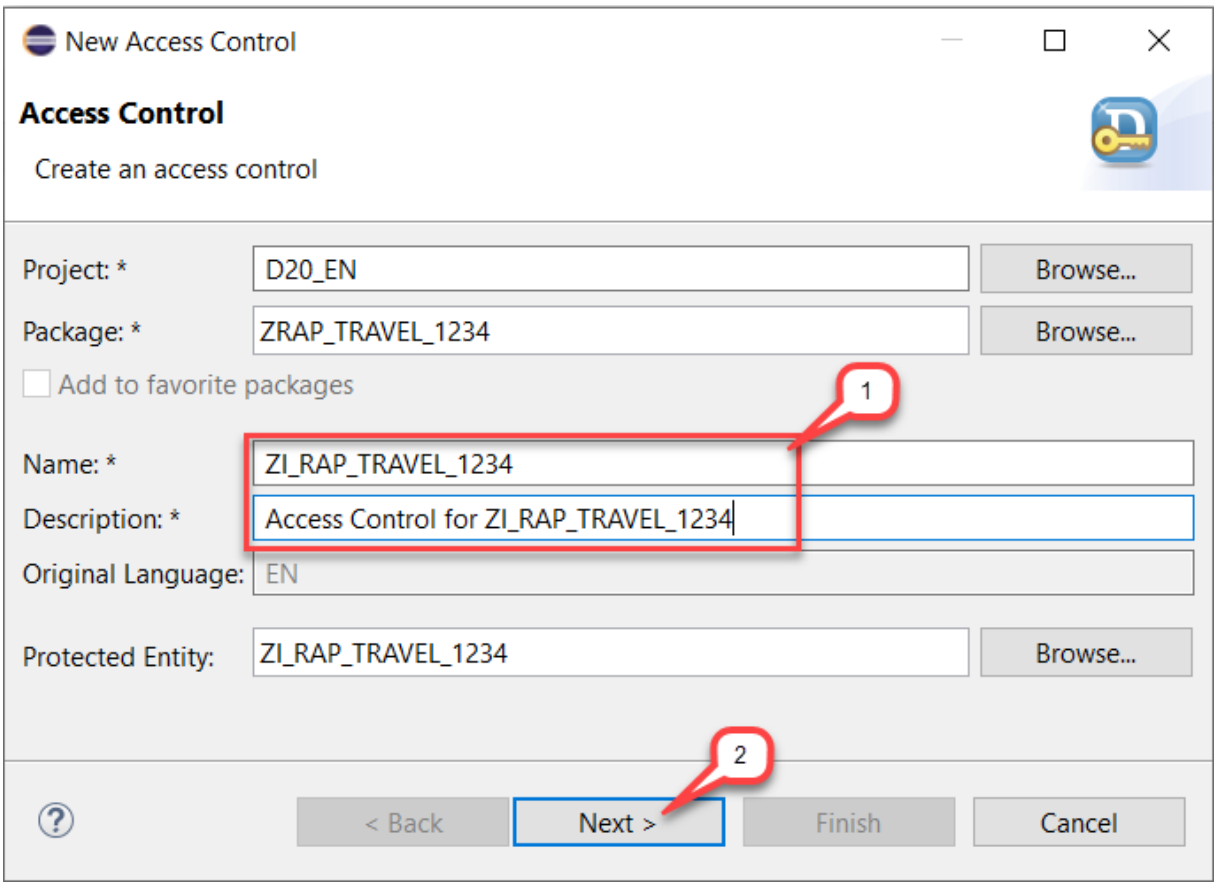


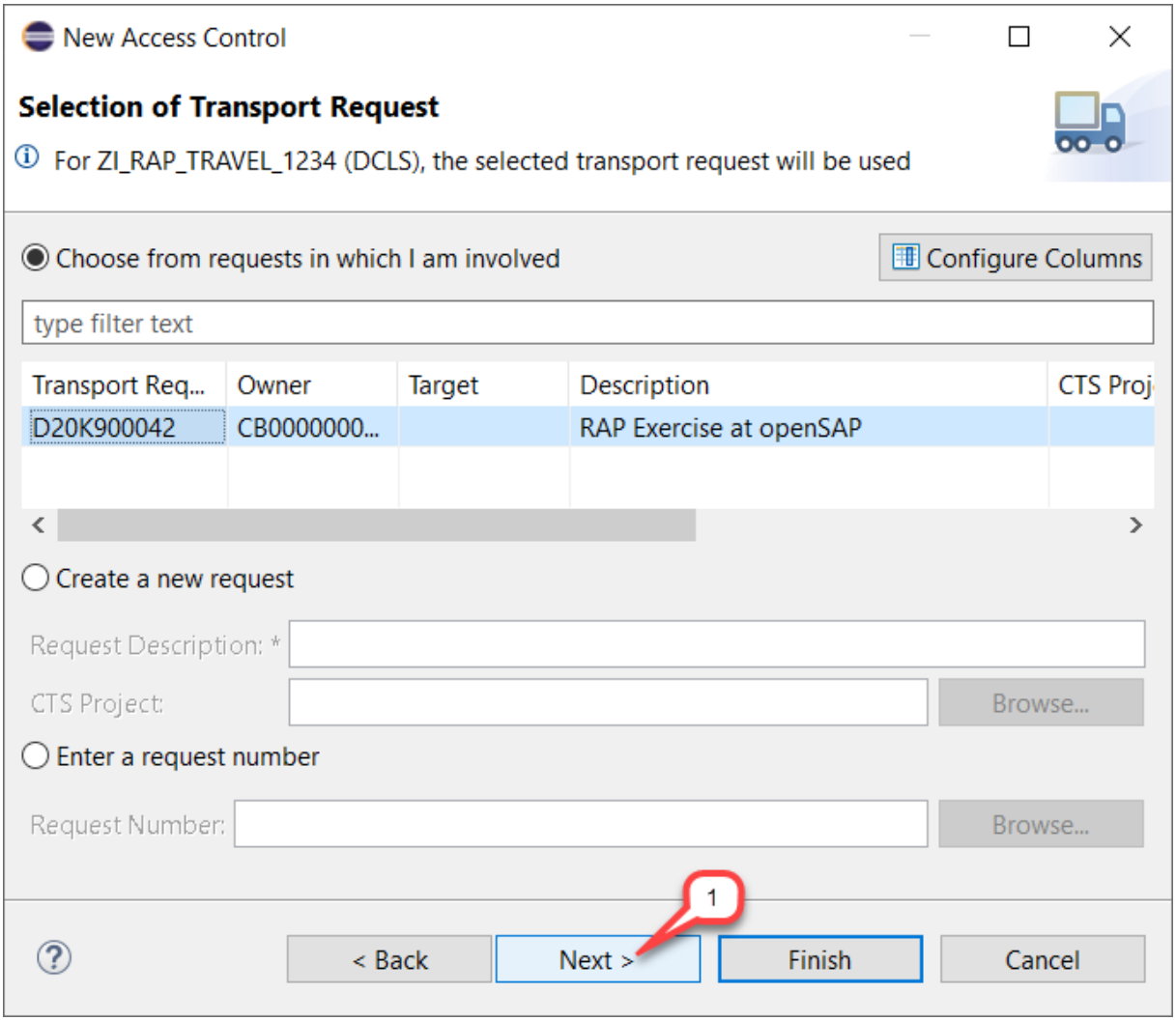
Étape 4 – Créer le CDS Access Control pour la vue BO ZI_RAP_Travel_####

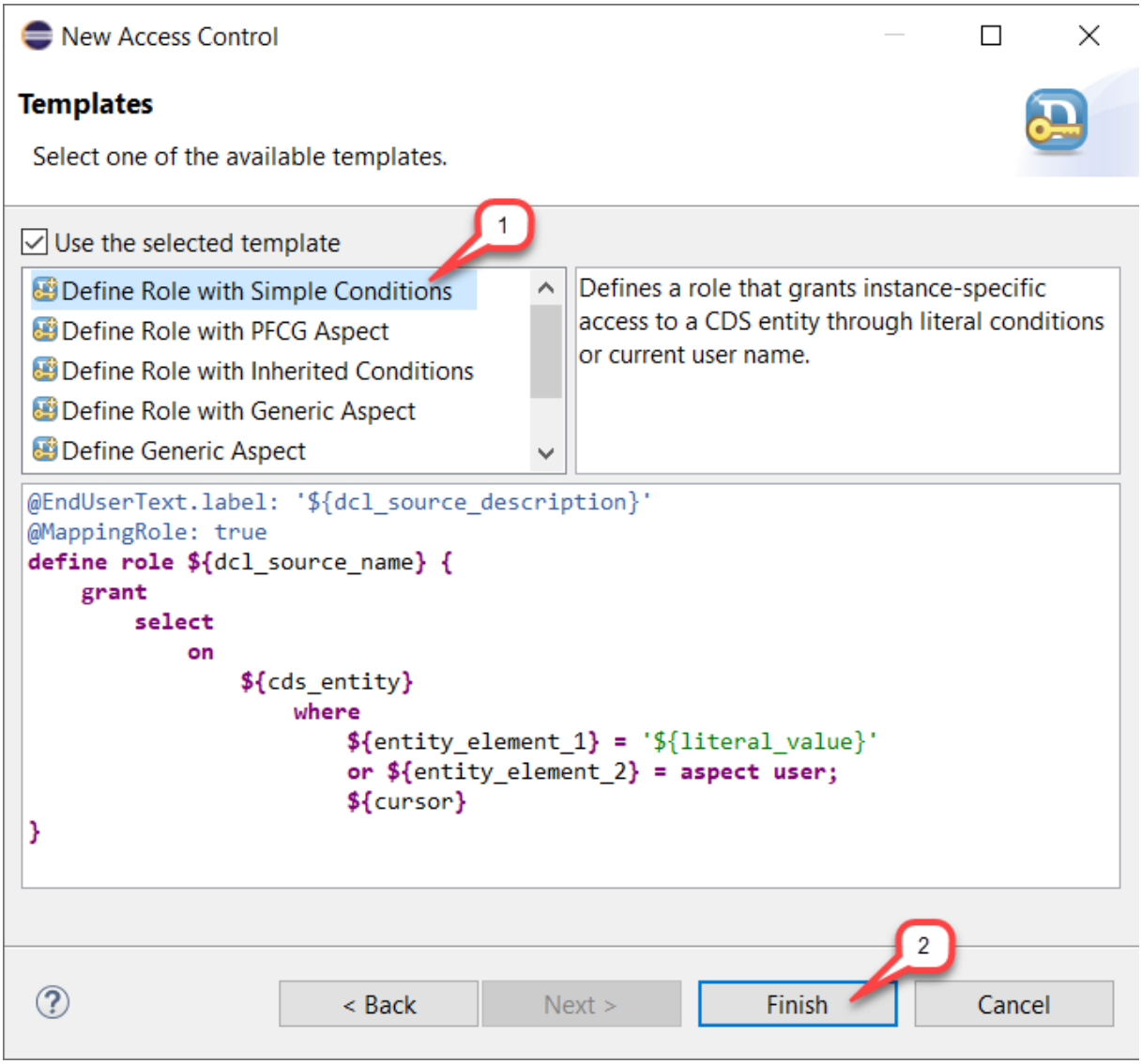
Création

- Clic droit sur ZI_RAP_Travel_#### > New Access Control
- Nom : ZI_RAP_Travel_####
- Modèle : Define Role with Simple Conditions


ZRAP_TRAVEL_1234 (20) Greenfield Implementation - Travel
Authorizations (4)
Business Services (2)
Cloud Communication Management (1)
Cloud Identity and Access Management (2)
Core Data Services (6)
Data Definitions (4)
ZC_RAP_BOOKING_1234
ZC_RAP_TRAVEL_1234
ZI_RAP_BOOKING_1234
ZI_RAP_TRAVEL_1234
Metadata Extensions (2)
Dictionary (3)
OAuth 2.0 Scopes (1)
Source Code Library (1)
1
2
New Data Definition
New Access Control
New Metadata Extension
New Service Definition
New ABAP Test Class
New Behavior Definition
Duplicate...
New Knowledge Transfer Document


New Access Control
Access Control
Create an access control
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
1
Name: *
ZI_RAP_TRAVEL_1234
Description: *
Access Control for ZI_RAP_TRAVEL_1234
Original Language:
EN
Protected Entity:
ZI_RAP_TRAVEL_1234
Browse...
2
< Back
Next >
Finish
Cancel


New Access Control
Selection of Transport Request
For ZI_RAP_TRAVEL_1234 (DCLS), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel



Remplacement de la clause WHERE

Remplacez les conditions factices de la clause **WHERE** par une condition littérale sur l'élément **CurrencyCode** autorisant uniquement la récupération des enregistrements dont le code de devise est « **EUR** ».

Utilisez l'extrait de code suivant :

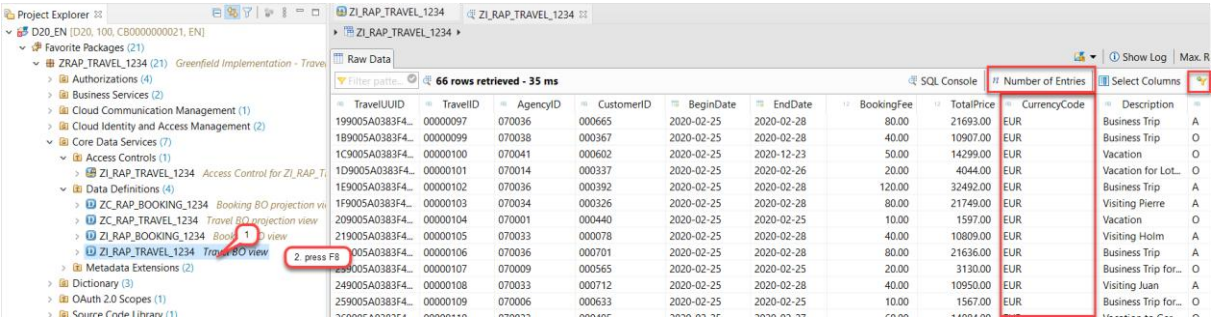
```
CurrencyCode = 'EUR';
```

Enregistrez et activez le contrôle d'accès.

Vérification

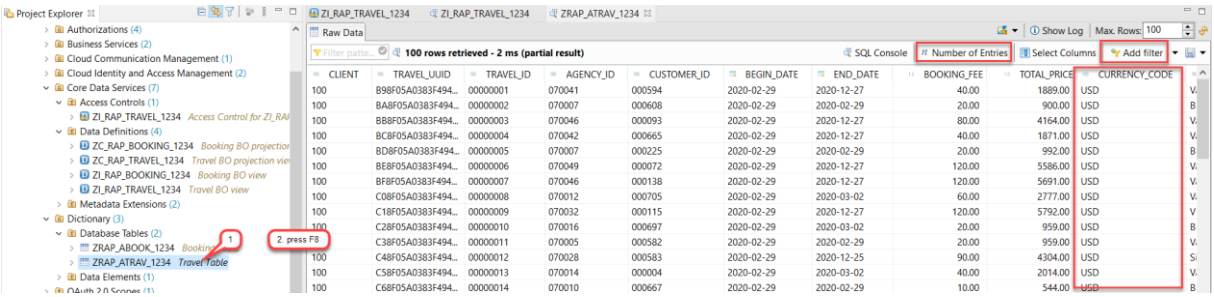
Pour vérifier le résultat :

- Sélectionnez la vue CDS ZI_RAP_TRAVEL_#### dans l'explorateur de projets
- Cliquez sur **F8** ou faites **clic droit > Open with > Data Preview**



Résultat : seuls les enregistrements avec la clé de devise EUR (EURO) sont récupérés de la base de données.

Vous pouvez comparer ce résultat avec l'aperçu des données de la table de base de données sous-jacente ZRAP_ATRAV_#### qui comprend également d'autres devises (par exemple USD, JPY).



Ajout d'une règle PFCG

Revenez maintenant au contrôle d'accès **CDS ZI_RAP_TRAVEL_####** et améliorez la règle d'accès avec une condition PFCG sur l'élément **TravelStatus**.
Seules les données autorisées selon l'objet d'autorisation **ZOSTAT####** doivent être récupérées et affichées pour chaque utilisateur.

Remplacez la clause WHERE par :

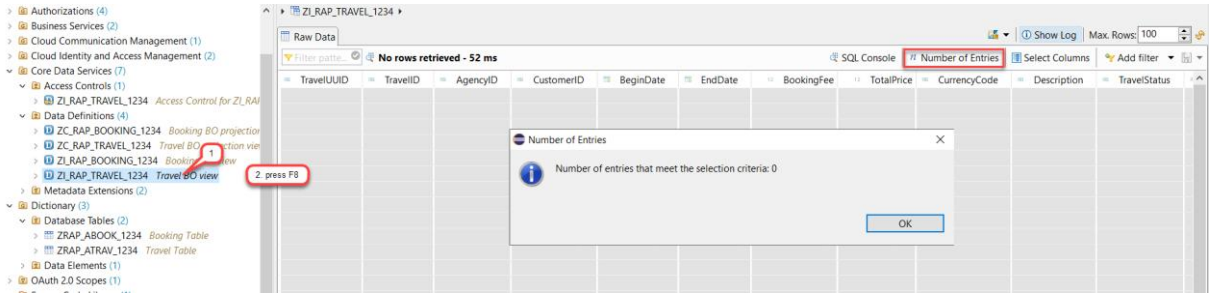
```
( TravelStatus )
                = aspect pfcg_auth ( ZOSTAT####, ZOSTAT####,  actvt = '03')
                 and
                 CurrencyCode = 'EUR';
```

Remplacez toutes les occurrences de #### par ton suffixe habituel.

Enregistrez et activer les changements.

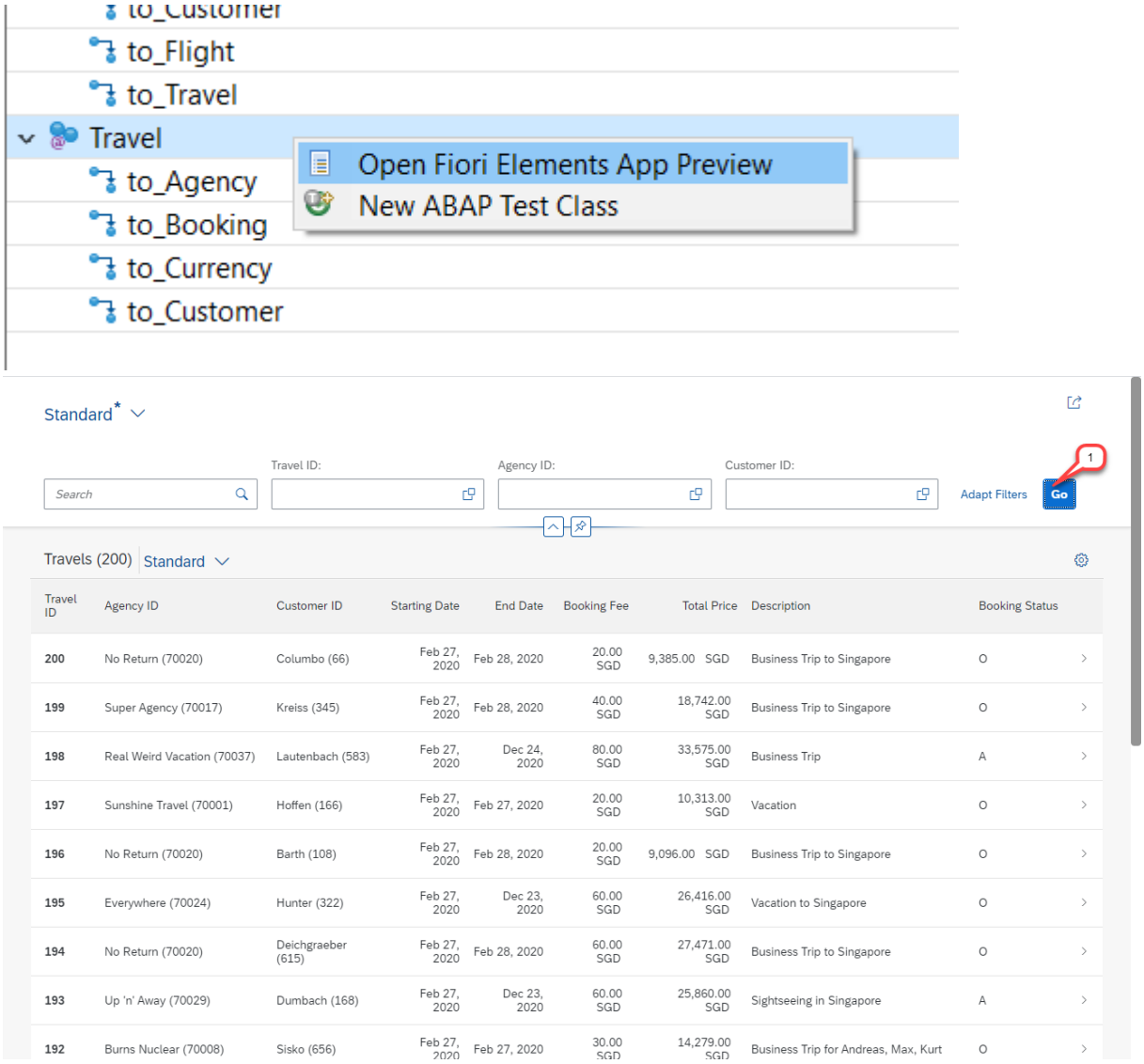
Vérification

Vérifiez maintenant à nouveau le résultat en lançant l'aperçu des données de la vue **ZI_RAP_TRAVEL_####.**



Aucune donnée n'est récupérée cette fois-ci
Cela est normal : ton utilisateur ne dispose pas encore des autorisations requises (objet ZOSTAT####).

Relance l'aperçu de l'application Travel via **la liaison de service** ou actualise la page (**F5**) dans le navigateur, puis clique sur **Go** pour récupérer les données.



Les données sont toujours récupérées.
Pourquoi ?
Parce que tu n'as pas encore défini de règles d'accès pour la vue de projection Travel BO ZC_RAP_TRAVEL_#### utilisée dans la définition de service.

Vous pouvez le vérifier en lançant aussi l'aperçu des données sur la CDS ZC_RAP_TRAVEL_####, et constater que les données sont visibles.

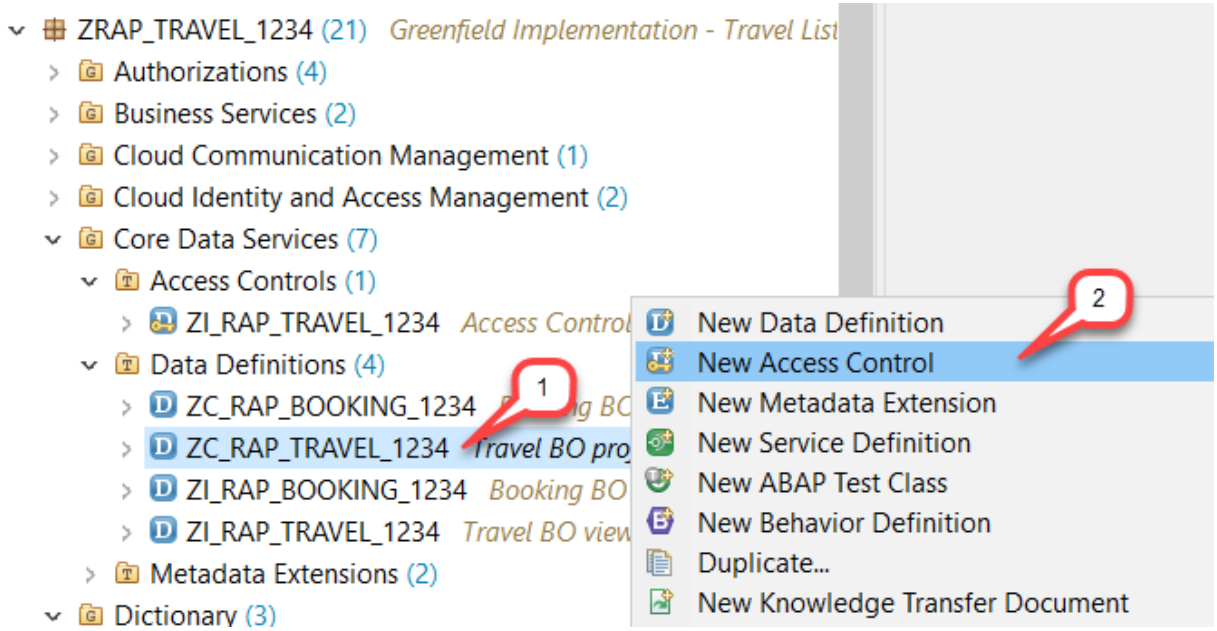
Un contrôle d'accès CDS doit être explicitement défini pour chaque entité CDS.
Il n'existe aucun héritage implicite des règles d'accès.

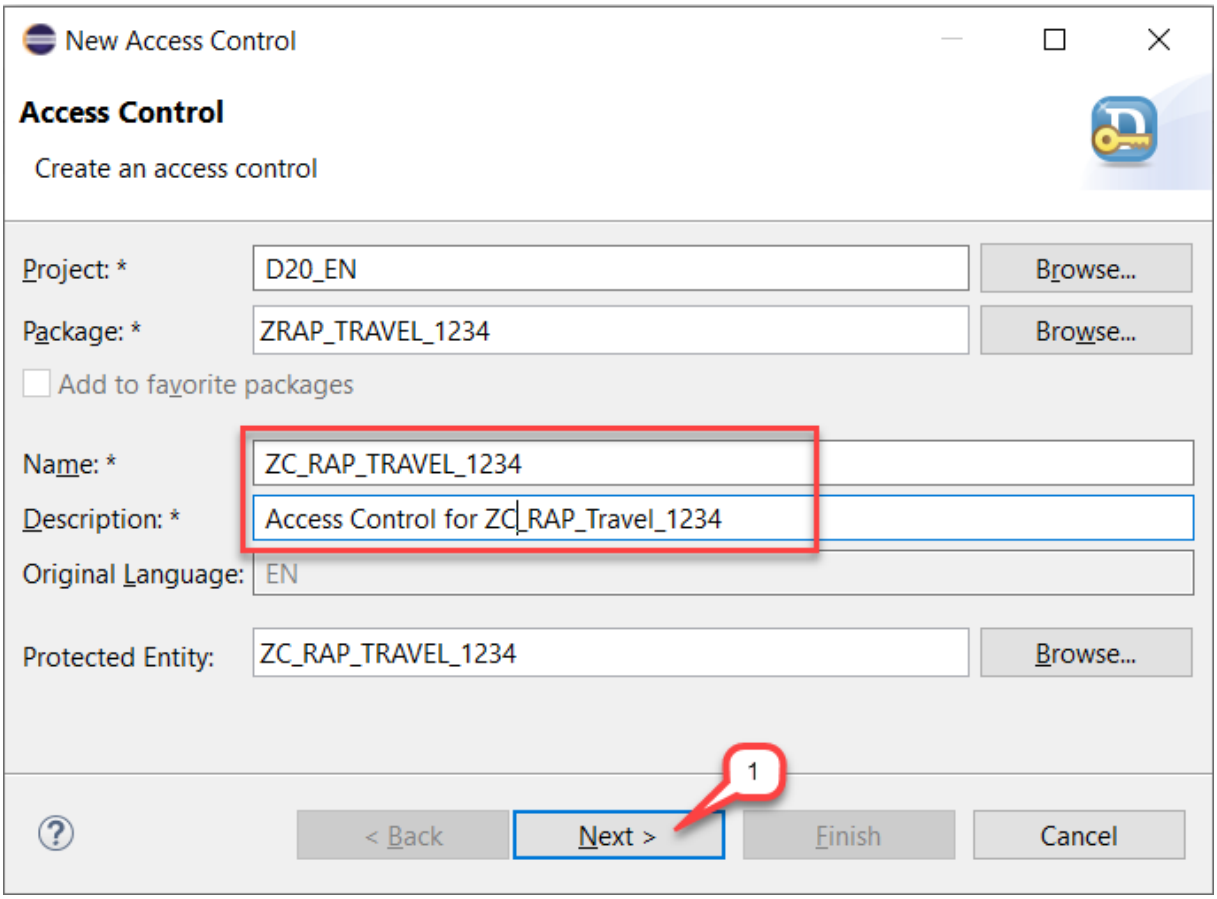
Étape 5. Créer un contrôle d'accès CDS pour la vue de projection BO de voyage et vérifier le résultat

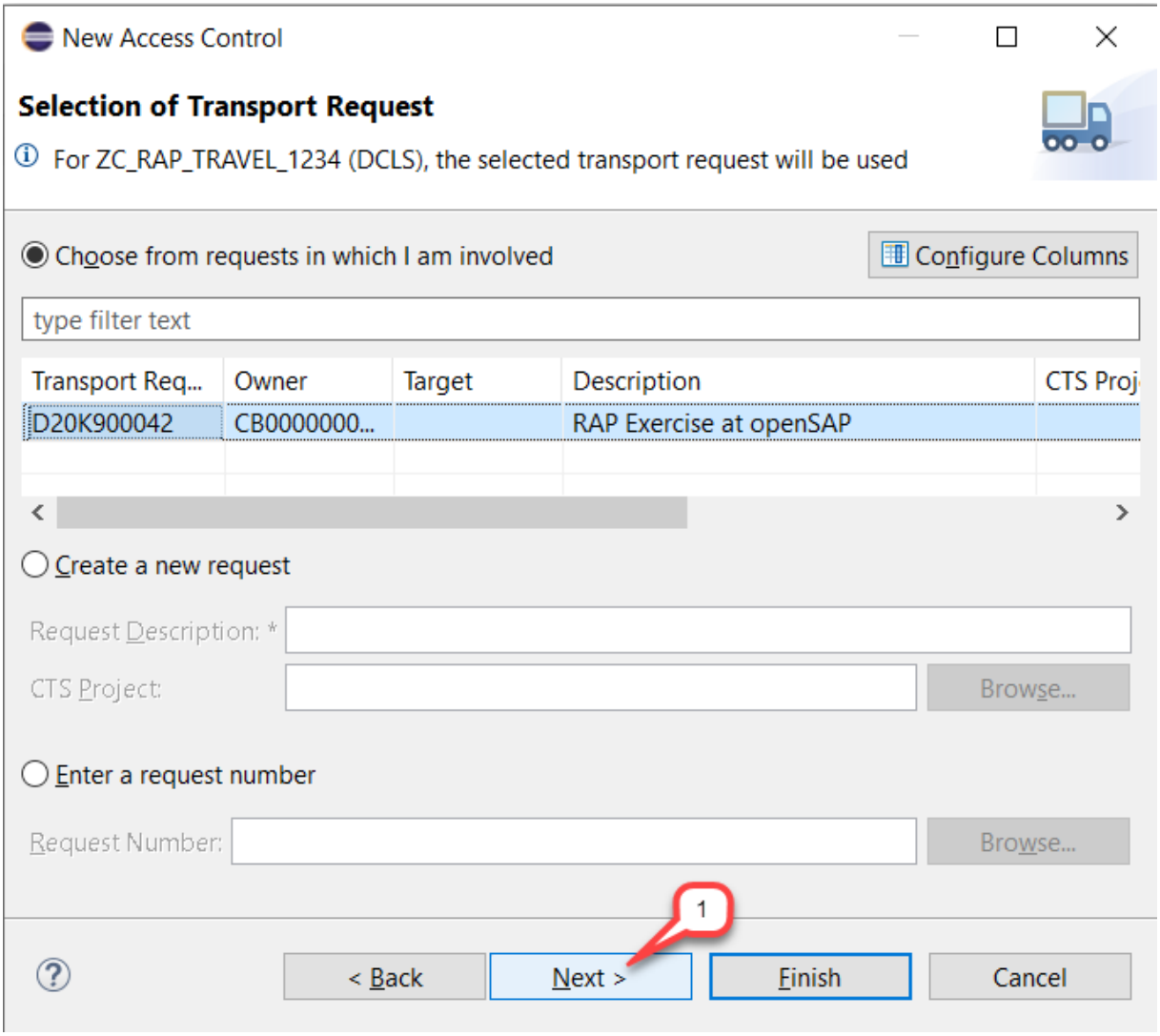
Vous allez maintenant définir le contrôle d'accès **CDS ZC_RAP_TRAVEL_####** pour la vue de projection **Travel BO** en héritant des règles d'accès définies dans la vue Travel BO sous-jacente.

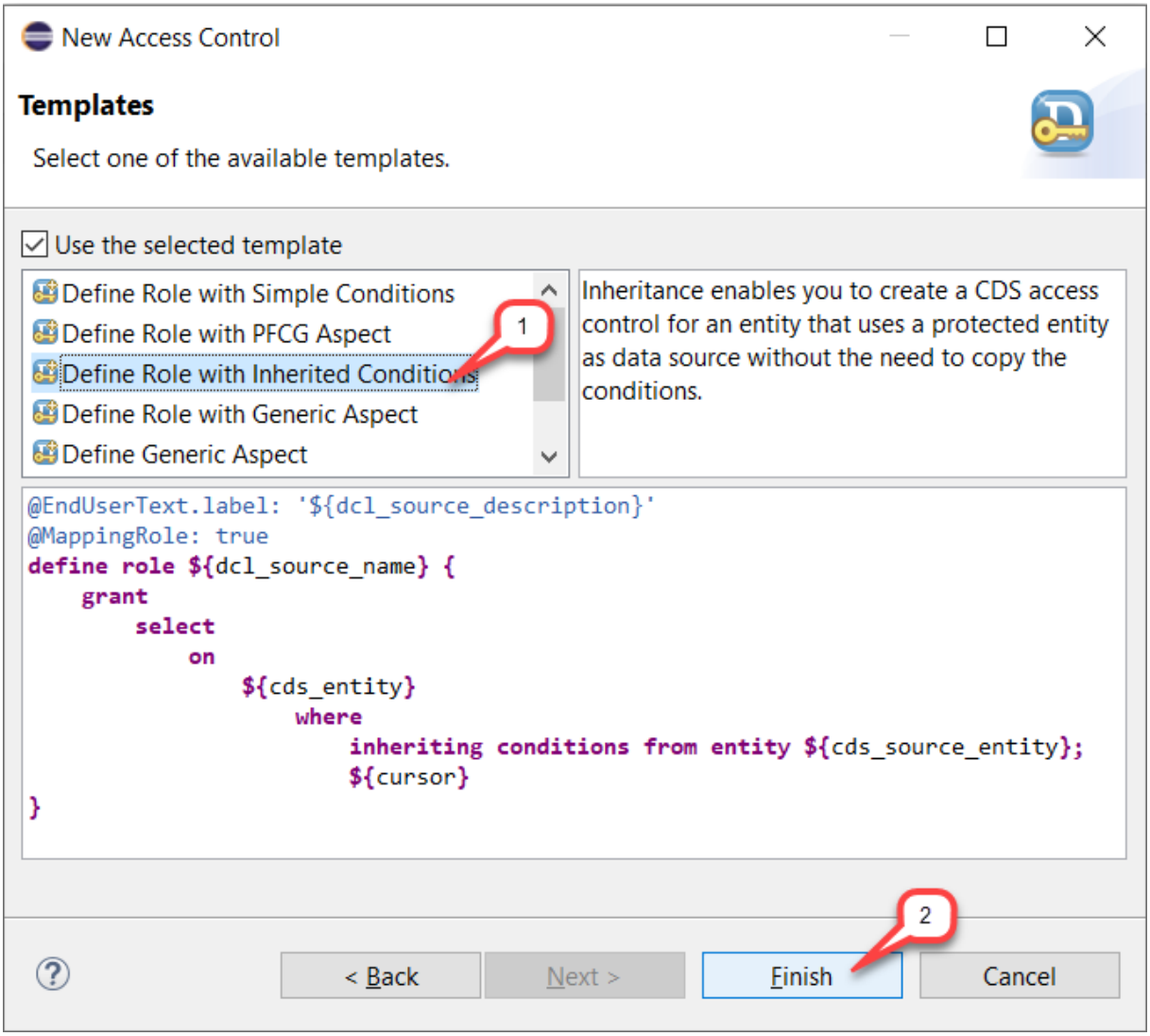
Étapes :

- Clic droit sur la vue **ZC_RAP_TRAVEL_####** dans l'explorateur de projets
- Sélectionne **New Access Control**
- Nom : ZC_RAP_TRAVEL_####
- Description : *Access control for ZC_RAP_TRAVEL_####*
- **Suivant** >
- Affecte la requête de transport
- **Suivant** >
- Sélectionne : **Define Role with inherited Conditions**
- **Terminer**


ZRAP_TRAVEL_1234 (21) Greenfield Implementation - Travel List
Authorizations (4)
Business Services (2)
Cloud Communication Management (1)
Cloud Identity and Access Management (2)
Core Data Services (7)
Access Controls (1)
ZI_RAP_TRAVEL_1234 Access Control
Data Definitions (4)
ZC_RAP_BOOKING_1234
ZC_RAP_TRAVEL_1234 Travel BO proj
ZI_RAP_BOOKING_1234 Booking BO
ZI_RAP_TRAVEL_1234 Travel BO view
Metadata Extensions (2)
Dictionary (3)
1
2
New Data Definition
New Access Control
New Metadata Extension
New Service Definition
New ABAP Test Class
New Behavior Definition
Duplicate...
New Knowledge Transfer Document


New Access Control
Access Control
Create an access control
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
Name: *
ZC_RAP_TRAVEL_1234
Description: *
Access Control for ZC_RAP_Travel_1234
Original Language:
EN
Protected Entity:
ZC_RAP_TRAVEL_1234
Browse...
1
< Back
Next >
Finish
Cancel


New Access Control
Selection of Transport Request
For ZC_RAP_TRAVEL_1234 (DCLS), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
1
< Back
Next >
Finish
Cancel



Le contrôle d'accès CDS est créé et s'ouvre dans l'éditeur.

Brève explication :

- L'annotation **@MappingRole: true** est définie en haut.
- Le rôle CDS est défini avec **DEFINE ROLE**.
- **GRANT SELECT ON** indique la vue protégée.
- **inheriting conditions from entity** définit de quelle entité les règles sont héritées.

Remplace l'entité fictive **cds_source_name** par :

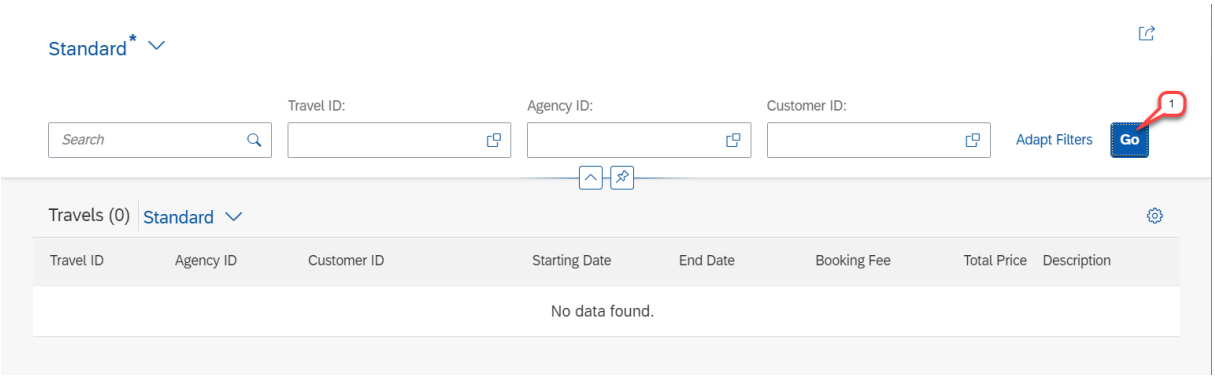
```
*ZC_RAP_TRAVEL_1234
1  @EndUserText.label: 'Access Control for ZC_RAP_Travel_1234'
2  @MappingRole: true
3  define role ZC_RAP_TRAVEL_1234 {
4      grant
5          select
6              on
7                  ZC_RAP_Travel_1234
8                      where
9                          inheriting conditions from entity ZI_RAP_Travel_1234;
10
11 }
```

Enregistre et active les changements.

Vérification

Lance l'aperçu des données de la **vue ZC_RAP_TRAVEL_####** dans ADT
ou actualise l'application **Travel** dans le navigateur.

Aucune donnée ne devrait être récupérée cette fois-ci.
Pourquoi ? Les règles de sécurité de ZI_RAP_... sont désormais appliquées via héritage.



Étape 6. Solution de contournement : ajuster les règles d'accès au CDS

Comme mentionné précédemment, le cours n'aborde pas la création des modèles d'autorisation PFCG pour accorder les autorisations à l'utilisateur courant.

Solution :

Pour récupérer un accès complet aux données, modifie le rôle ZI_RAP_TRAVEL_#### :

```
ZI_RAP_TRAVEL_1234
1  @EndUserText.label: 'Access Control for ZI_RAP_TRAVEL_1234'
2  @MappingRole: true
3  define role ZI_RAP_TRAVEL_1234 {
4      grant
5          select
6              on
7                  ZI_RAP_Travel_1234
8                      where
9  //                      ( TravelStatus )
10 //                        = aspect pfcg_auth ( ZOSTAT1234, ZOSTAT1234,  actvt = '03')
11 //                          and
12 //                          CurrencyCode = 'EUR';
13                         true;
14
15 }
```

(commente les conditions existantes dans **WHERE** et remplace par **true;**)

Enregistre et active les changements.

Vous pouvez maintenant :

- Visualiser les données via l'aperçu **ZI_RAP_TRAVEL_####**
- Tester à nouveau l'application Fiori Travel dans le navigateur

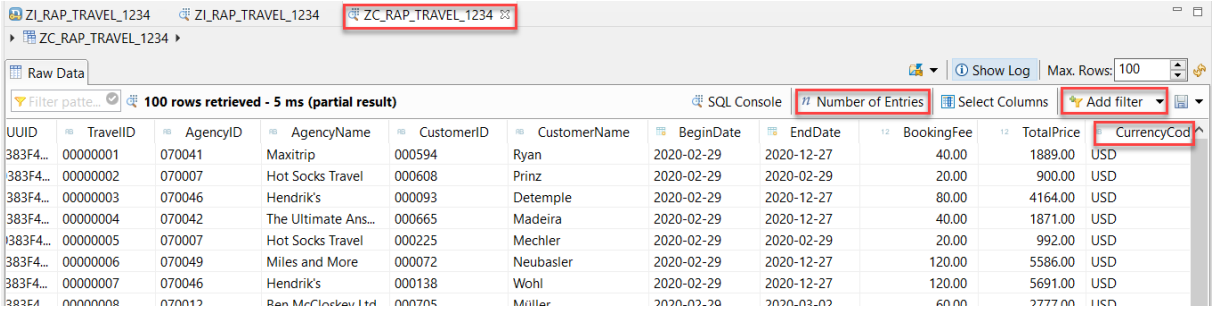
ZI_RAP_TRAVEL_1234 | ZI_RAP_TRAVEL_1234 | ZI_RAP_TRAVEL_1234

Raw Data | Show Log | Max. Rows: 100

Filter patte... | 100 rows retrieved - 16 ms (partial result) | SQL Console | Number of Entries | Select Columns | Add filter

TravelUUID	TravelID	AgencyID	CustomerID	BeginDate	EndDate	BookingFee	TotalPrice	CurrencyCode	Description	Trave
B98F05A0383F4...	00000001	070041	000594	2020-02-29	2020-12-27	40.00	1889.00	USD	Vacation	O
BA8F05A0383F4...	00000002	070007	000608	2020-02-29	2020-02-29	20.00	900.00	USD	Business Trip for...	O
BB8F05A0383F4...	00000003	070046	000093	2020-02-29	2020-12-27	80.00	4164.00	USD	Vacation	O
BC8F05A0383F4...	00000004	070042	000665	2020-02-29	2020-12-27	40.00	1871.00	USD	Vacation	O
BD8F05A0383F4...	00000005	070007	000225	2020-02-29	2020-02-29	20.00	992.00	USD	Business Trip for...	O
BE8F05A0383F4...	00000006	070049	000072	2020-02-29	2020-12-27	120.00	5586.00	USD	Vacation	O
BF8F05A0383F4...	00000007	070046	000138	2020-02-29	2020-12-27	120.00	5691.00	USD	Vacation	O
C08F05A0383F4...	00000008	070012	000705	2020-02-29	2020-03-02	60.00	2777.00	USD	Vacation for Lot...	O
C18F05A0383F4...	00000009	070032	000115	2020-02-29	2020-12-27	120.00	5792.00	USD	Visiting Anna	A
C28F05A0383F4...	00000010	070016	000697	2020-02-29	2020-03-02	20.00	959.00	USD	Business Trip to ...	O

Il n'est pas nécessaire de modifier le contrôle d'accès pour ZC_RAP_TRAVEL_####
Les conditions sont héritées automatiquement de ZI_RAP_....



Résumé

Dans cette unité, vous avez appris à :

- Créer des objets **d'autorisation PFCG** dans ABAP
- Implémenter un **CDS Access Control** avec :
  - Conditions littérales
  - Conditions PFCG
- Propager la restriction d'accès aux vues projetées avec **inheriting conditions from entity**
- Appliquer une solution temporaire avec **WHERE true** pour tests

4. Semaine 3 — *Activation du comportement transactionnel d'une application*

Cette semaine marque une **étape clé** dans la création de ton application RAP : après avoir modélisé les données et exposé le service OData, vous allez maintenant **rendre l'application interactive et modifiable**, c'est-à-dire **transactionnelle**.

Concrètement, vous allez :

- Définir le **comportement métier de base** de l'objet (Behavior Definition),
- Créer une **projection de comportement** pour l'exposer au service OData,
- Ajouter de la **logique applicative spécifique** (ex. : règles de validation),
- Utiliser le **langage EML** pour manipuler les entités dans le code ABAP,
- Implémenter les **méthodes de modification** (create, update, delete, etc.),
- Activer la **gestion des brouillons (draft)** pour permettre aux utilisateurs de travailler sur des données non encore sauvegardées.

Objectif final : Permettre à l'utilisateur d'interagir pleinement avec l'application Travel (ajouter, modifier, supprimer des voyages), comme dans une vraie application Fiori.

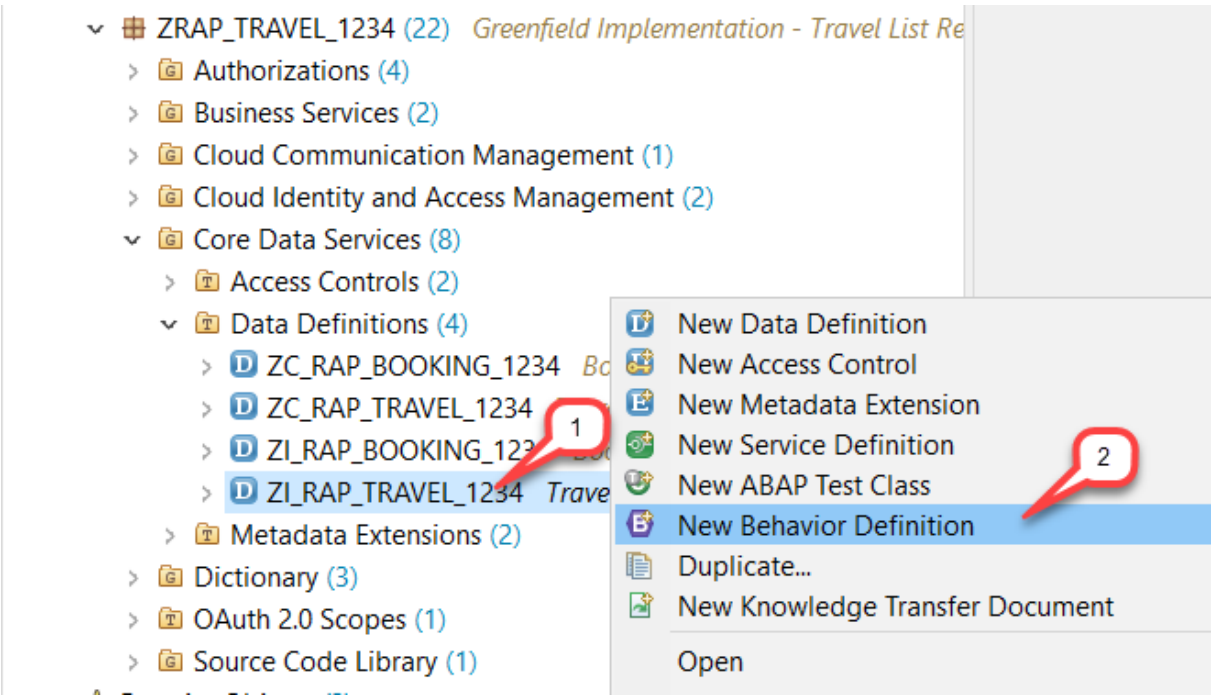
4.1 Définition du comportement basique de l'objet métier (Unité 1)

Objectif :

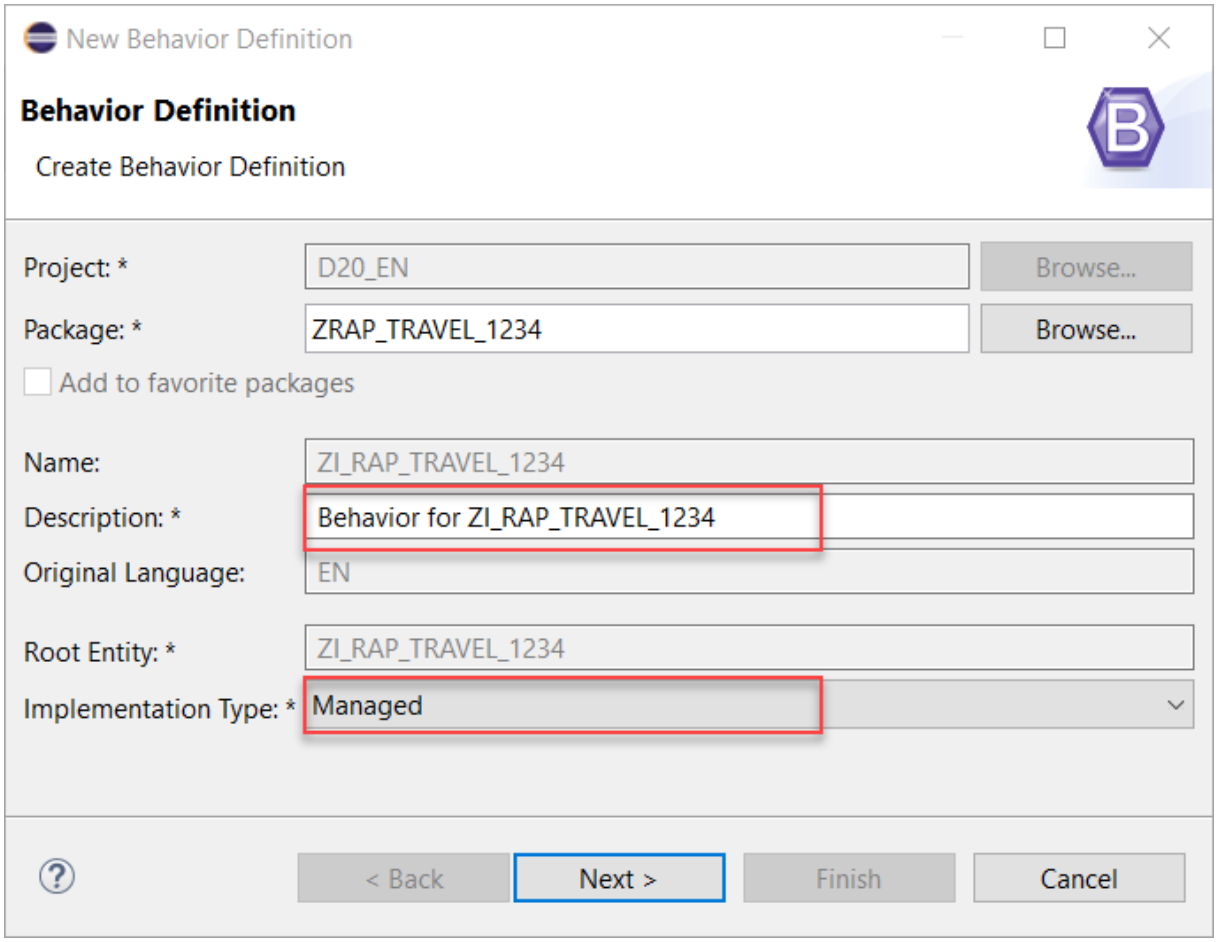
Définir les opérations transactionnelles de base (Créer, Mettre à jour, Supprimer) pour les entités **Travel** et **Booking**, et tester l'application SAP Fiori enrichie.

Étape 1 : Créer la définition de comportement (Behavior Definition)

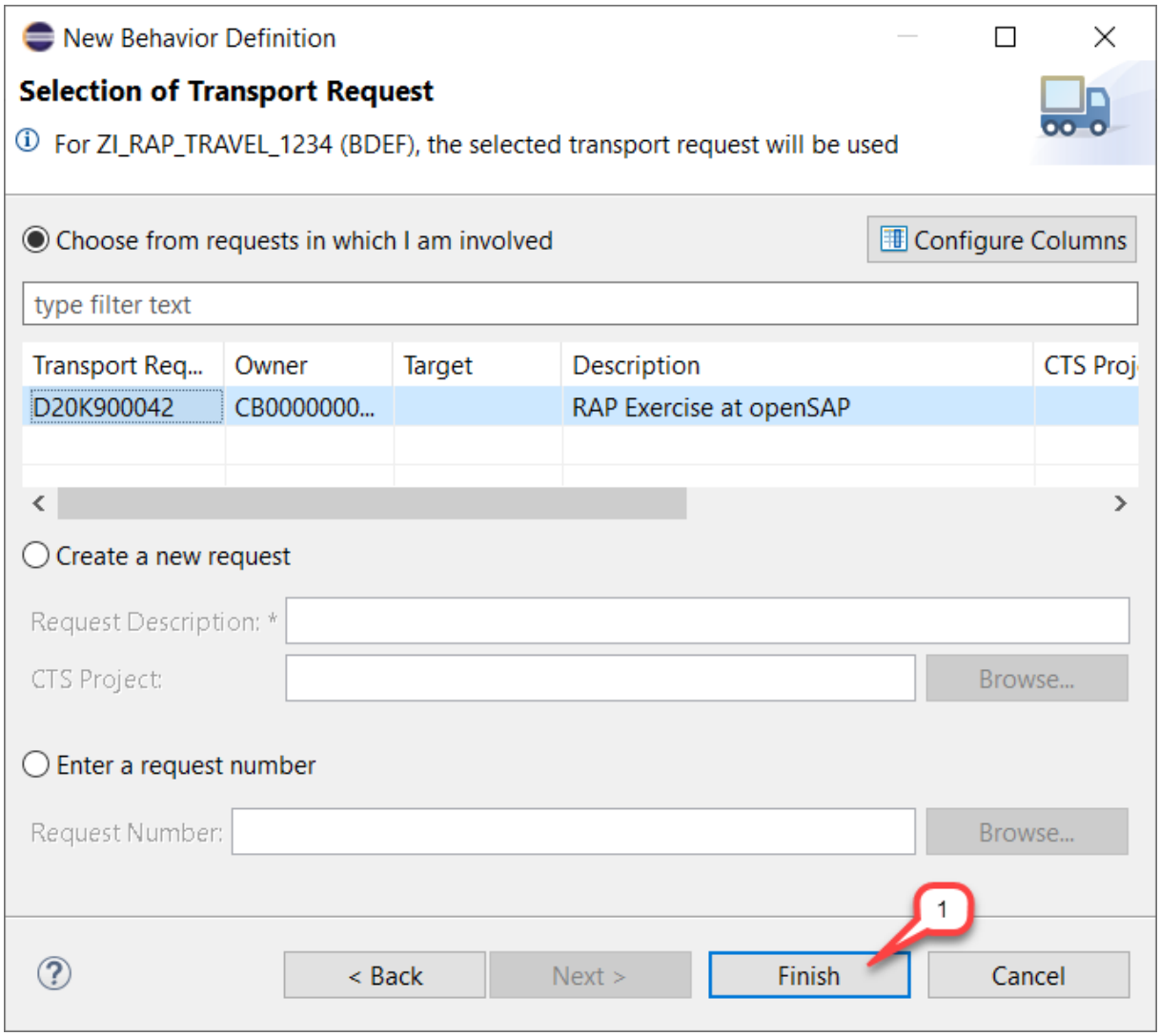
1. Clic droit sur la vue CDS racine ZI_RAP_TRAVEL_#### → **New Behavior Definition**



- Le nom doit être identique à celui de la vue CDS racine (non modifiable)
- Type d'implémentation : **Managed**


New Behavior Definition
Behavior Definition
Create Behavior Definition
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
Name:
ZI_RAP_TRAVEL_1234
Description: *
Behavior for ZI_RAP_TRAVEL_1234
Original Language:
EN
Root Entity: *
ZI_RAP_TRAVEL_1234
Implementation Type: *
Managed
< Back
Next >
Finish
Cancel

2. Assignez une demande de transport et terminez



- Un squelette est généré pour les entités Travel et Booking avec les opérations CUD (Create, Update, Delete)
- Associations définies (ex. : création d'une réservation depuis un voyage)

ZI_RAP_TRAVEL_1234

```
1  managed; // implementation in class zbp_i_rap_travel_1234 unique;
2
3  define behavior for ZI_RAP_Travel_1234 //alias <alias_name>
4  //persistent table <???>
5  //lock master
6  //authorization master ( instance )
7  //etag master <field_name>
8  {
9    create;
10   update;
11   delete;
12   association _Booking { create; }
13 }
14
15 define behavior for ZI_RAP_Booking_1234 //alias <alias_name>
16 //persistent table <???>
17 //lock dependent by <association>
18 //authorization dependent by <association>
19 //etag master <field_name>
20 {
21   update;
22   delete;
23 }
```

Étape 2 : Ajuster la définition de comportement pour l'entité Travel

- Alias Travel conservé

```
alias Travel
```

- Définir la table persistante zrap_atrav_####

```
persistent table zrap_atrav_####
```

- Activer le **verrouillage optimiste** (ETag) avec le champ LocalLastChangedAt

```
etag master LocalLastChangedAt
```

- Activer le verrouillage master (gestion générique par RAP)

```
lock master
```

- Clé primaire TravelUUID gérée automatiquement (numbering : managed, readonly)

```
field ( numbering : managed, readonly ) TravelUUID;
```

- Définir le **mapping** entre champs CDS et colonnes table

```
mapping for zrap_atrav_####
 {
  TravelUUID       = travel_uuid;
  TravelID         = travel_id;
  AgencyID         = agency_id;
  CustomerID       = customer_id;
  BeginDate        = begin_date;
  EndDate          = end_date;
  BookingFee       = booking_fee;
  TotalPrice       = total_price;
  CurrencyCode     = currency_code;
  Description      = description;
  TravelStatus     = overall_status;
  CreatedBy        = created_by;
  CreatedAt        = created_at;
  LastChangedBy    = last_changed_by;
  LastChangedAt    = last_changed_at;
  LocalLastChangedAt = local_last_changed_at;
 }
```

- Opérations autorisées : create, update, delete, association _Booking create

*ZI_RAP_TRAVEL_1234

```
1  managed; // implementation in class zbp_i_rap_travel_12[illegible]nique;
2
3  define behavior for ZI_RAP_Travel_1234 alias Travel
4  persistent table zrap_atrav_1234
5  lock master
6  //authorization master ( instance )
7  etag master LocalLastChangedAt
8  {
9    create;
10   update;
11   delete;
12   association _Booking { create; }
13
14   field ( numbering : managed, readonly ) TravelUUID;
15
16   mapping for zrap_atrav_1234
17   {
18     TravelUUID         = travel_uuid;
19     TravelID           = travel_id;
20     AgencyID           = agency_id;
21     CustomerID         = customer_id;
22     BeginDate          = begin_date;
23     EndDate            = end_date;
24     BookingFee         = booking_fee;
25     TotalPrice         = total_price;
26     CurrencyCode       = currency_code;
27     Description        = description;
28     TravelStatus       = overall_status;
29     CreatedBy          = created_by;
30     CreatedAt          = created_at;
31     LastChangedBy      = last_changed_by;
32     LastChangedAt      = last_changed_at;
33     LocalLastChangedAt = local_last_changed_at;
34   }
35 }
36
37 define behavior for ZI_RAP_Booking_1234 //alias <alias_name>
38 //persistent table <???>
```

Exemple résumé pour Travel :

```
define behavior for ZI_RAP_Travel_#### alias Travel

persistent table zrap_atrav_####

lock master

//authorization master ( instance )

etag master LocalLastChangedAt

{

  create;

  update;

  delete;

  association _Booking { create; }


  field ( numbering : managed, readonly ) TravelUUID;


  mapping for zrap_atrav_####

  {

    TravelUUID        = travel_uuid;

    TravelID          = travel_id;

    AgencyID          = agency_id;

    CustomerID        = customer_id;

    BeginDate         = begin_date;

    EndDate           = end_date;

    BookingFee        = booking_fee;

    TotalPrice        = total_price;

    CurrencyCode      = currency_code;
```

```
  Description       = description;
  TravelStatus      = overall_status;
  CreatedBy         = created_by;
  CreatedAt         = created_at;
  LastChangedBy     = last_changed_by;
  LastChangedAt     = last_changed_at;
  LocalLastChangedAt = local_last_changed_at;
 }
}
```

Enregistrez **mais** n'activez pas les changements.

Étape 3 : Ajuster la définition de comportement pour l'entité Booking

- Alias Booking conservé

```
alias Booking
```

- Table persistante zrap_abook_####

```
persistent table zrap_abook_####
```

- Verrouillage dépendant de _Travel (root entity)

```
lock dependent by _Travel
```

- ETag LocalLastChangedAt activé aussi pour Booking

```
etag master LocalLastChangedAt
```

- Association explicite _Travel déclarée

```
 association _Travel;
```

- Clé primaire BookingUUID gérée automatiquement (numbering : managed, readonly)

```
 field ( numbering : managed, readonly ) BookingUUID;
```

- TravelUUID défini en lecture seule (readonly) car utilisé dans l'association

```
field ( readonly ) TravelUUID;
```

- Opérations autorisées : update, delete (pas create direct ici, création via association Travel)
- Mapping entre CDS et table défini

```
mapping for zrap_abook_####
{
  BookingUUID       = booking_uuid;
  TravelUUID        = travel_uuid;
  BookingID         = booking_id;
  BookingDate       = booking_date;
  CustomerID        = customer_id;
  CarrierID         = carrier_id;
  ConnectionID      = connection_id;
  FlightDate        = flight_date;
  FlightPrice       = flight_price;
  CurrencyCode      = currency_code;
  CreatedBy         = created_by;
  LastChangedBy     = last_changed_by;
  LocalLastChangedAt = local_last_changed_at;
}
```

*ZI_RAP_TRAVEL_1234

```
1  managed; // implementation in class zbp_i_rap_travel_1234 unique;
2
3  define behavior for ZI_RAP_Travel_1234 alias Travel
4  persistent table zrap_atrav_1234
5  lock m
36
37 define behavior for ZI_RAP_Booking_1234 alias Booking
38 persistent table zrap_abook_1234
39 lock dependent by _Travel
40 //authorization dependent by <association>
41 etag master LocalLastChangedAt
42 {
43   association _Travel;
44
45   update;
46   delete;
47
48   field ( numbering : managed, readonly ) BookingUUID;
49   field ( readonly ) TravelUUID;
50
51   mapping for zrap_abook_1234
52   {
53     BookingUUID        = booking_uuid;
54     TravelUUID         = travel_uuid;
55     BookingID          = booking_id;
56     BookingDate        = booking_date;
57     CustomerID         = customer_id;
58     CarrierID          = carrier_id;
59     ConnectionID       = connection_id;
60     FlightDate         = flight_date;
61     FlightPrice        = flight_price;
62     CurrencyCode       = currency_code;
63     CreatedBy          = created_by;
64     LastChangedBy      = last_changed_by;
65     LocalLastChangedAt = local_last_changed_at;
66   }
67 }
68
```

1 2 3 4 6 7 8 9

Exemple résumé pour Booking :

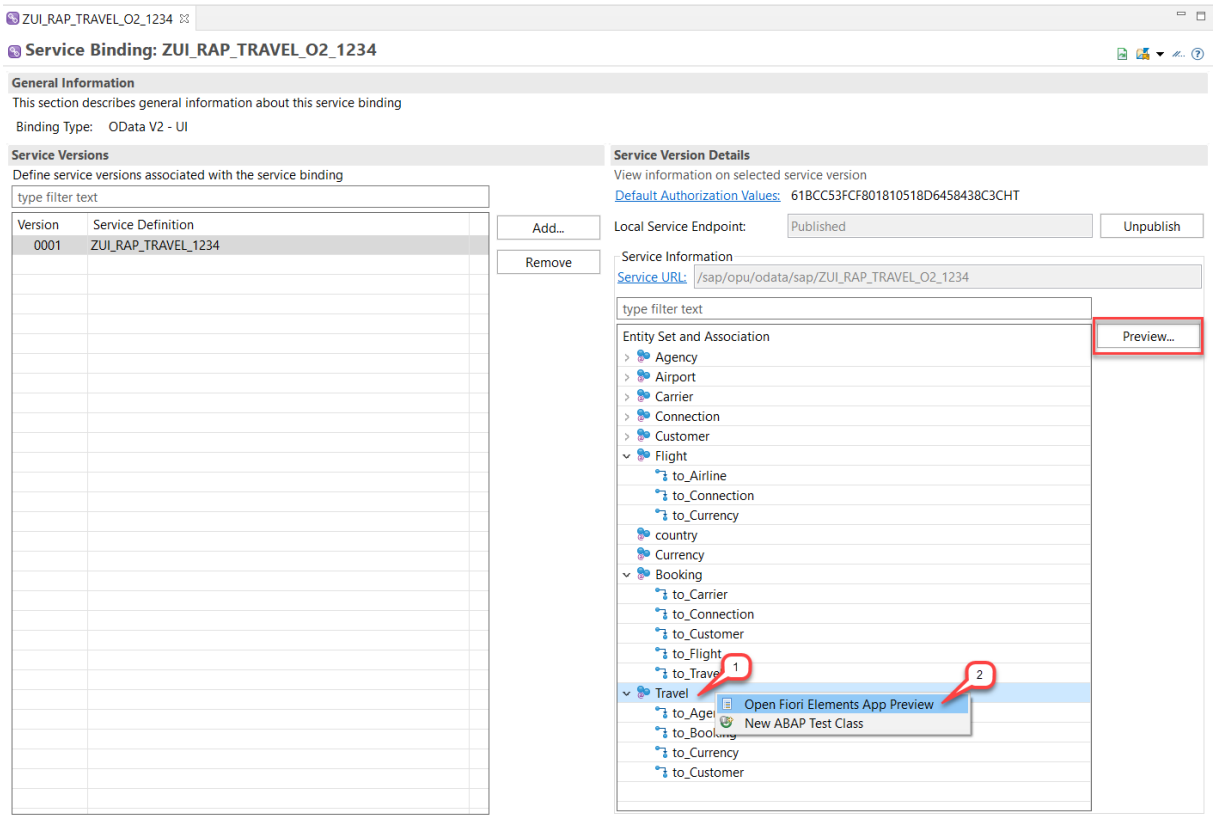
```
define behavior for ZI_RAP_Booking_#### alias Booking

persistent table zrap_abook_####

lock dependent by _Travel

//authorization dependent by

etag master LocalLastChangedAt

{

 association _Travel;

 update;

 delete;

 field ( numbering : managed, readonly ) BookingUUID;

 field ( readonly ) TravelUUID;

 mapping for zrap_abook_####

 {

  BookingUUID       = booking_uuid;

  TravelUUID        = travel_uuid;

  BookingID         = booking_id;

  BookingDate       = booking_date;

  CustomerID        = customer_id;

  CarrierID         = carrier_id;

  ConnectionID      = connection_id;

  FlightDate        = flight_date;

  FlightPrice       = flight_price;

  CurrencyCode      = currency_code;

  CreatedBy         = created_by;
```

```
  LastChangedBy      = last_changed_by;

  LocalLastChangedAt = local_last_changed_at;

}

}
```

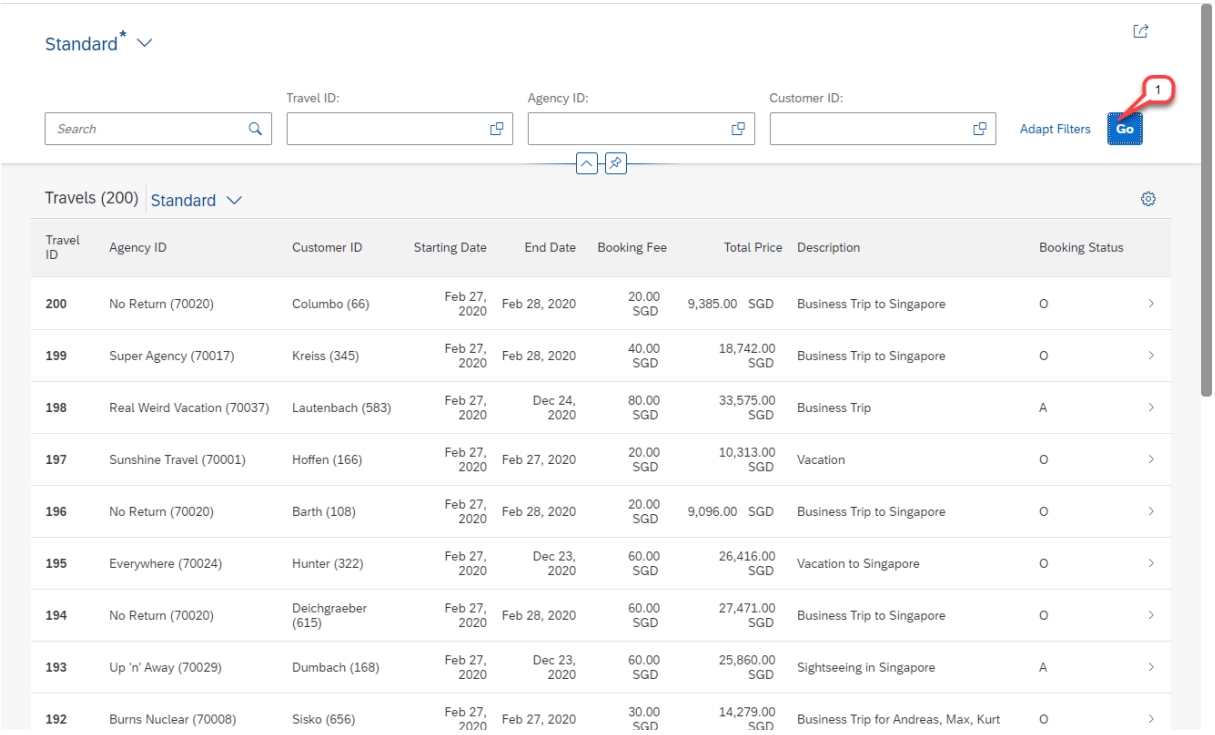
Maintenant, enregistrez et activez les modifications.

Étape 4 : Prévisualiser l'application Travel

- Ouvrez la liaison de service ZUI_RAP_TRAVEL_O2_####
- Lancez l'application via le menu contextuel sur l'entité Travel → **Open Fiori Elements App preview**



- La liste s'affiche, mais toujours en lecture seule car le comportement n'est pas encore projeté



Travel ID	Agency ID	Customer ID	Starting Date	End Date	Booking Fee	Total Price	Description	Booking Status
200	No Return (70020)	Columbo (66)	Feb 27, 2020	Feb 28, 2020	20.00 SGD	9,385.00 SGD	Business Trip to Singapore	O
199	Super Agency (70017)	Kreiss (345)	Feb 27, 2020	Feb 28, 2020	40.00 SGD	18,742.00 SGD	Business Trip to Singapore	O
198	Real Weird Vacation (70037)	Lautenbach (583)	Feb 27, 2020	Dec 24, 2020	80.00 SGD	33,575.00 SGD	Business Trip	A
197	Sunshine Travel (70001)	Hoffen (166)	Feb 27, 2020	Feb 27, 2020	20.00 SGD	10,313.00 SGD	Vacation	O
196	No Return (70020)	Barth (108)	Feb 27, 2020	Feb 28, 2020	20.00 SGD	9,096.00 SGD	Business Trip to Singapore	O
195	Everywhere (70024)	Hunter (322)	Feb 27, 2020	Dec 23, 2020	60.00 SGD	26,416.00 SGD	Vacation to Singapore	O
194	No Return (70020)	Deichgraeber (615)	Feb 27, 2020	Feb 28, 2020	60.00 SGD	27,471.00 SGD	Business Trip to Singapore	O
193	Up 'n' Away (70029)	Dumbach (168)	Feb 27, 2020	Dec 23, 2020	60.00 SGD	25,860.00 SGD	Sightseeing in Singapore	A
192	Burns Nuclear (70008)	Sisko (656)	Feb 27, 2020	Feb 27, 2020	30.00 SGD	14,279.00 SGD	Business Trip for Andreas, Max, Kurt	O

Résumé final

- Création des définitions de comportement pour les entités Travel et Booking
- Définition des opérations transactionnelles de base (CRUD)
- Configuration du verrouillage optimiste (ETag) et du verrouillage maître/dépendant
- Mapping des données persistantes
- Test via prévisualisation de l'application

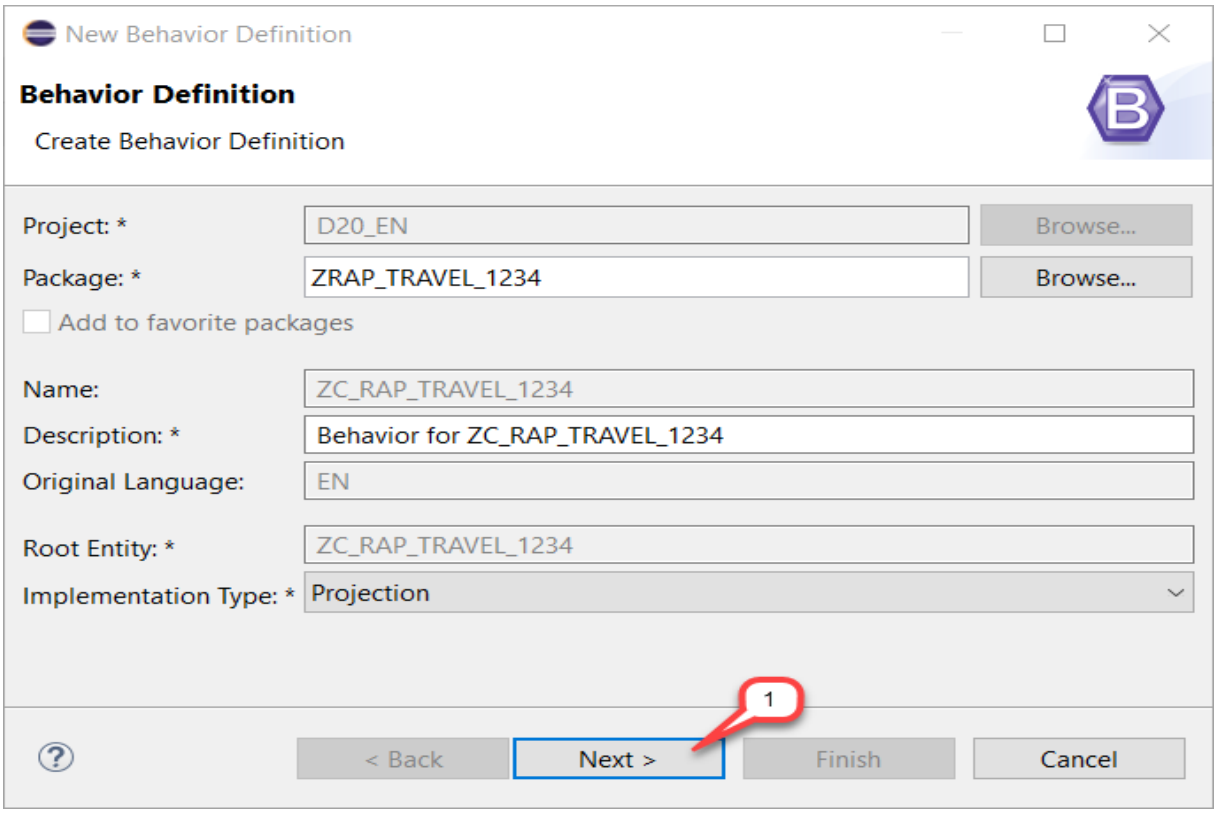
4.2 Création de la projection de comportement de l'objet métier (Unité 2)

Objectif :

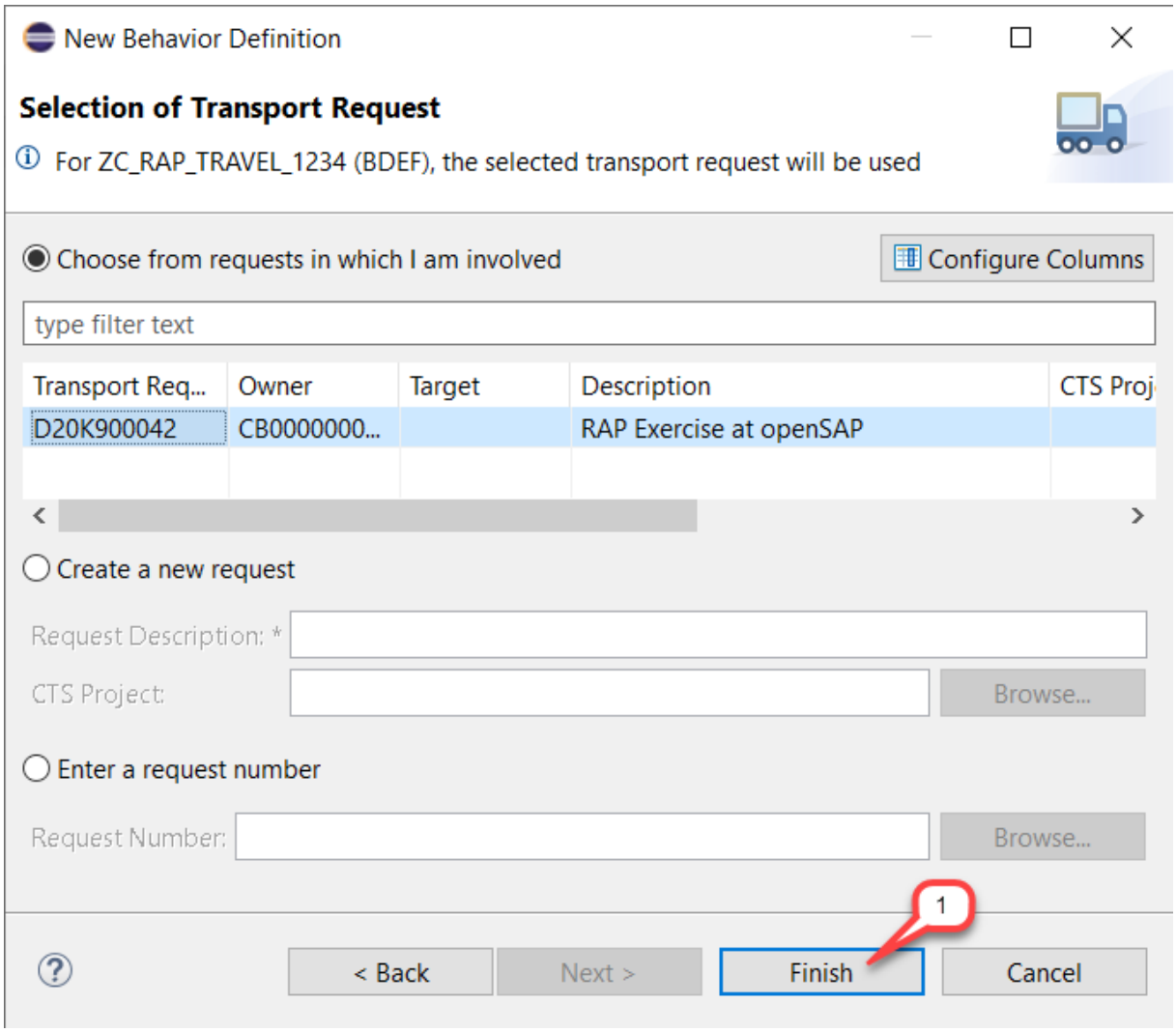
Projeter les fonctionnalités transactionnelles (create, update, delete) définies précédemment, pour les rendre **accessibles via l'application Fiori**.

Étape 1 : Créer la Behavior Projection

- Clic droit sur la vue **ZC_RAP_TRAVEL_####** (projection CDS de la vue BO) → **New Behavior Definition**
- Nom généré automatiquement (doit être le même que la vue CDS : ZC_RAP_TRAVEL_####)
- Type d'implémentation : **Projection**



- Assignez une demande de transport → **Finish**



- Un squelette de projection est généré pour :
  - ZC_RAP_Travel_####
  - ZC_RAP_Booking_####

ZC_RAP_TRAVEL_1234

```
projection;

define behavior for ZC_RAP_Travel_1234 //alias <alias_name>
{
  use create;
  use update;
  use delete;

  use association _Booking { create; }
}

define behavior for ZC_RAP_Booking_1234 //alias <alias_name>
{
  use update;
  use delete;

  use association _Travel;
}
```

Explication :

- Le mot-clé **projection;** est utilisé en haut
- Toutes les opérations définies dans la behavior definition d'origine sont exposées avec le mot-clé **use**

Étape 2 : Ajuster la Behavior Projection

- **Travel entity**
  - Définir un **alias** :

    ```
    alias Travel
    ```

  - Ajouter la prise en charge du verrouillage optimiste :

    ```
    use etag
    ```

- **Booking entity**
  - Alias et verrouillage optimiste :

```
alias Booking
```

- Ajouter la prise en charge du verrouillage optimiste :

```
use etag
```

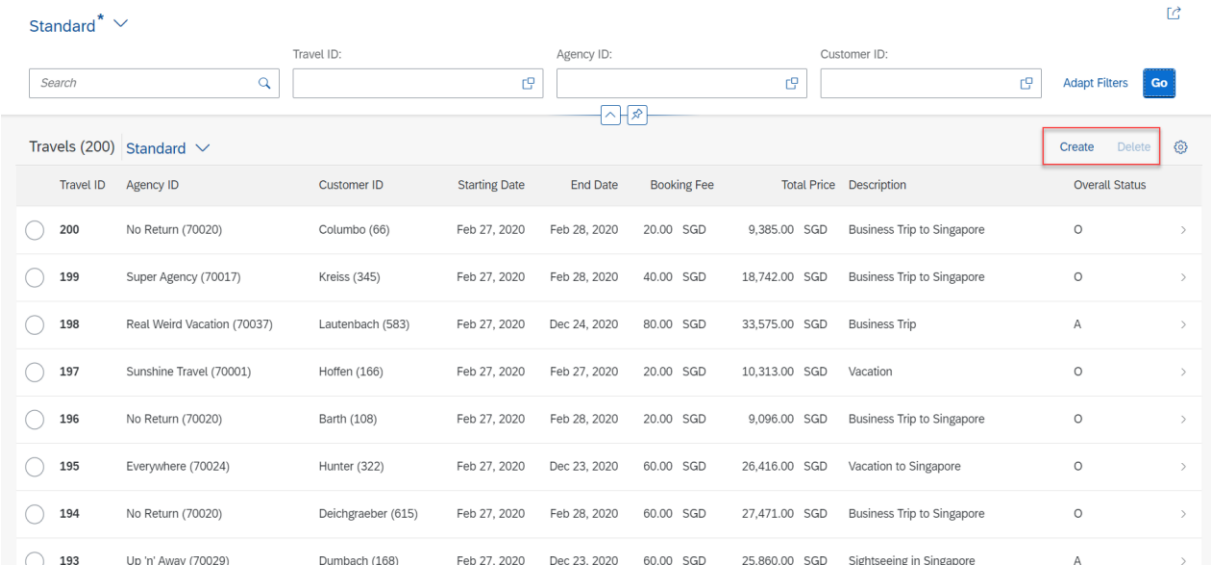
Code complet :

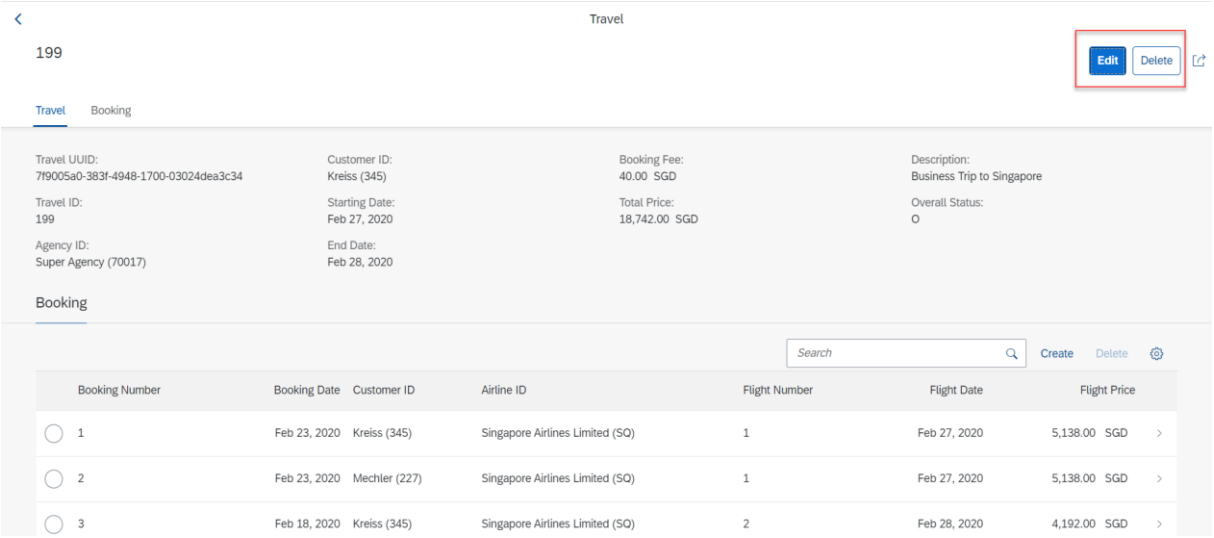
```
projection;


define behavior for ZC_RAP_Travel_#### alias Travel
use etag
{
  use create ;
  use update;
  use delete ;

  use association _Booking { create; }
 }

define behavior for ZC_RAP_Booking_#### alias Booking
use etag
{
 Use update ;
 Use delete;
 use associatio _Travel ;
}
```

Enregistrez et activez la projection de comportement.

Étape 3 : Prévisualiser l'application

- Ouvrez la liaison de service : ZUI_RAP_TRAVEL_O2_####
- Rafraîchissez l'application dans le navigateur (ou clic droit > **Open Fiori Elements App Preview**)
- Connectez-vous si besoin, puis cliquez sur **Go**
- Cette fois, les actions **Créer**, **Modifier**, **Supprimer** sont visibles et actives





Tester l'application

- Créez un voyage (Travel) avec des réservations (Booking)
- Modifiez les informations
- Supprimez un enregistrement
- Le bouton **Edit** est disponible sur chaque page objet

Résumé final

Vous avez appris à :

- Créer une **projection de comportement** (behavior projection) à partir de la définition managée
- Exposer les opérations CUD et les associations
- Activer le **verrou optimiste (ETag)** dans la projection
- Tester les capacités transactionnelles dans l'application Fiori

4.3 Compréhension du langage de manipulation d'entités (EML) (Unité 3)

Objectif :

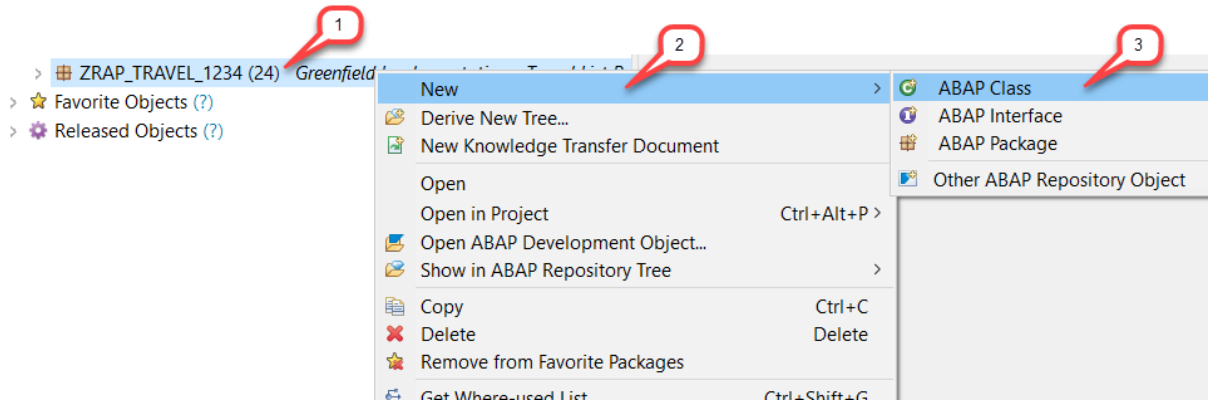
Apprendre à utiliser le **Entity Manipulation Language (EML)** pour manipuler un objet métier RAP (**Read, Create, Update, Delete**) en **ABAP**, en dehors de l'UI Fiori.
Cette syntaxe est essentielle pour :

- Les **validations**
- Les **determinations**
- Les **actions**
- Les tests unitaires

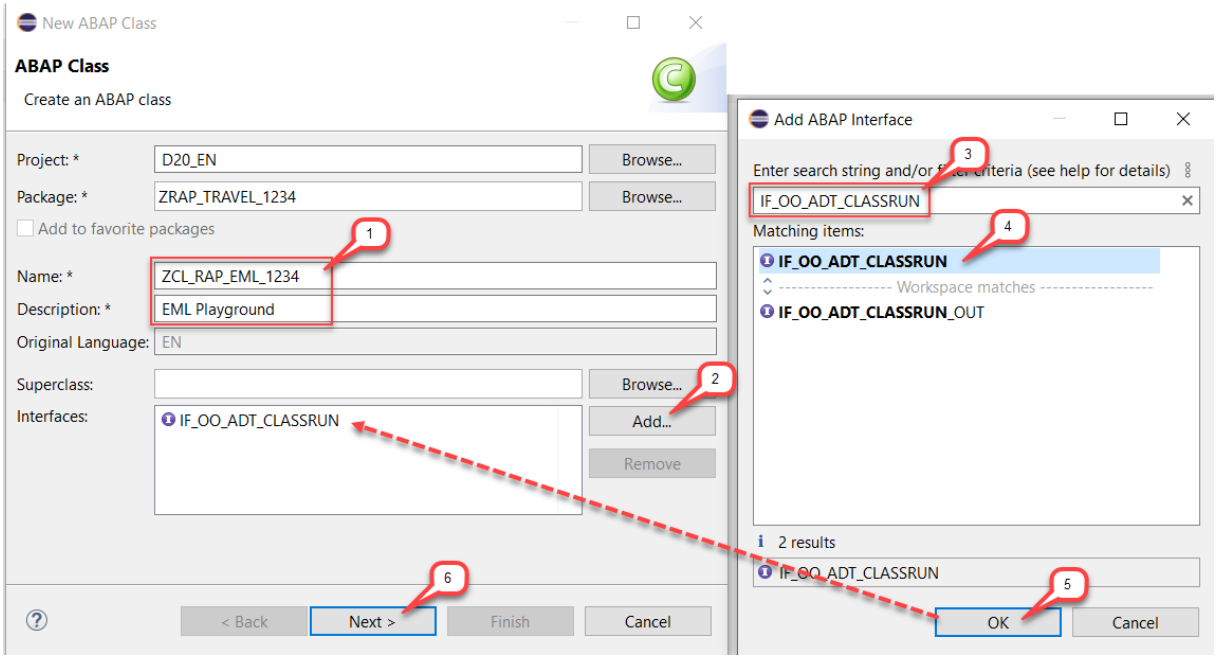
Étapes pratiques :

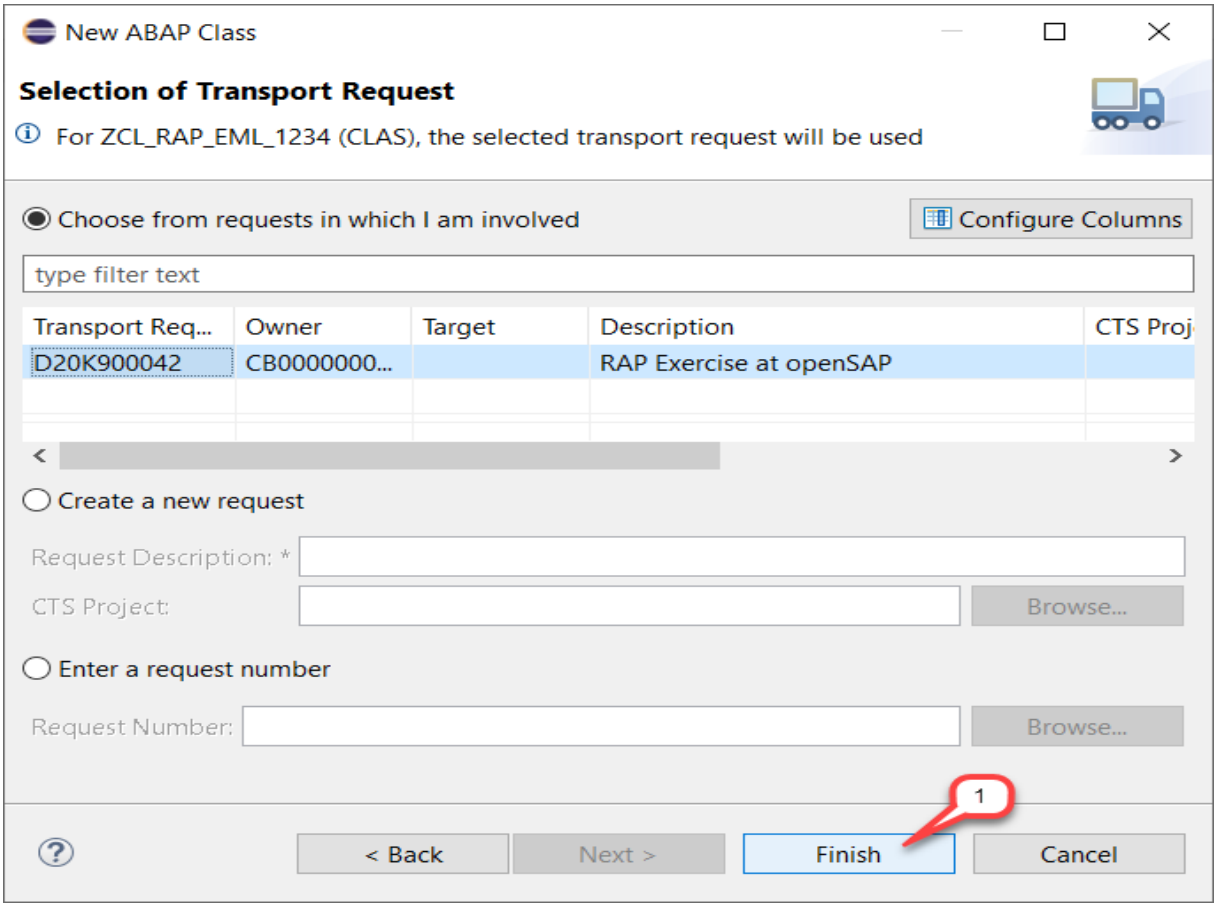
Étape 1 – Créer la classe de test

- Créez une classe ZCL_RAP_EML_#### dans le package ZRAP_TRAVEL_####



- Implémentez l'interface : if_oo_adt_classrun


New ABAP Class
ABAP Class
Create an ABAP class
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
Name: *
ZCL_RAP_EML_1234
Description: *
EML Playground
Original Language:
EN
Superclass:
Browse...
Interfaces:
IF_OO_ADT_CLASSRUN
Add...
Remove
< Back
Next >
Finish
Cancel
Add ABAP Interface
Enter search string and/or filter criteria (see help for details)
IF_OO_ADT_CLASSRUN
Matching items:
IF_OO_ADT_CLASSRUN
Workspace matches
IF_OO_ADT_CLASSRUN_OUT
2 results
IF_OO_ADT_CLASSRUN
OK
Cancel
1
2
3
4
5
6


New ABAP Class
Selection of Transport Request
For ZCL_RAP_EML_1234 (CLAS), the selected transport request will be used
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Browse...
Enter a request number
Request Number:
Browse...
< Back
Next >
Finish
Cancel
1

ZCL_RAP_EML_1234

ZCL_RAP_EML_1234

```
1  CLASS zcl_rap_eml_1234 DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5
6    PUBLIC SECTION.
7
8      INTERFACES if_oo_adt_classrun .
9    PROTECTED SECTION.
10   PRIVATE SECTION.
11 ENDCLASS.
12
13
14
15 CLASS zcl_rap_eml_1234 IMPLEMENTATION.
16 ENDCLASS.
```

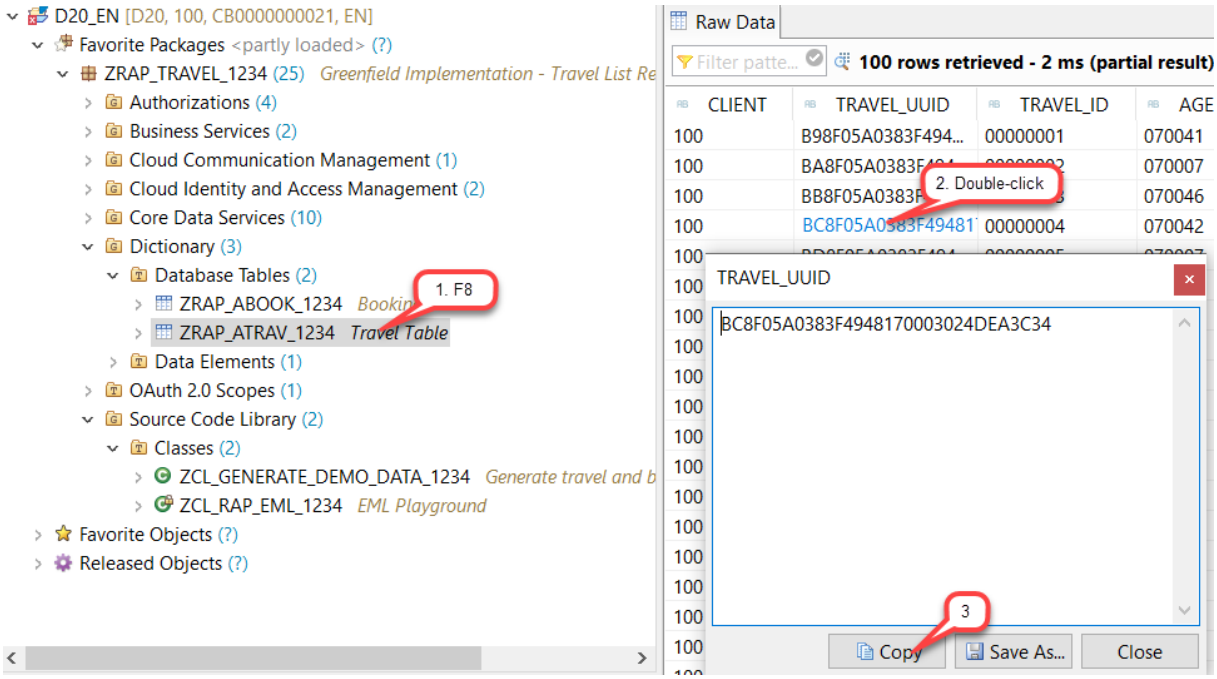
Opérations EML :

Étape 2 – Lecture

2.1 Lire une entité par clé :

```
" step 1 - READ

  READ ENTITIES OF ZI_RAP_Travel_####

    ENTITY travel

      FROM VALUE #( ( TravelUUID = '<your uuid>' ) )

    RESULT DATA(travels).



  out->write( travels ).
```

```
15 CLASS zcl_rap_eml_1234 IMPLEMENTATION.
16   METHOD if_oo_adt_classrun~main.
17     " step 1 - READ
18     READ ENTITIES OF ZI_RAP_Travel_1234
19           ENTITY travel
20             FROM VALUE #( ( TravelUUID = '<your uuid>' ) )
21           RESULT DATA(travels).
22
23     out->write( travels ).
24
25   ENDMETHOD.
26
27 ENDCLASS.
```




```
14
15 CLASS zcl_rap_eml_1234 IMPLEMENTATION.
16   METHOD if_oo_adt_classrun~main.
17     " step 1 - READ
18     READ ENTITIES OF ZI_RAP_Travel_1234
19           ENTITY travel
20             FROM VALUE #( ( TravelUUID = 'BC8F05A0383F4948170003024DEA3C34' ) )
21           RESULT DATA(travels).
22
23     out->write( travels ).
24
25   ENDMETHOD.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'**application *console***.

L'opération **READ** est exécutée et le résultat s'affiche dans la console.
Seul le champ clé est renseigné. Ceci est dû à l'absence de spécification des champs à lire.
Les clés sont fournies par défaut.

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Console

ABAP Console

Table

TRAVELUUID	TRAVELID	AGENCYID	CUSTOMERID	BEGINDATE	ENDDATE	BOOKINGFEE	TOTALPRICE	CURRENCYCODE	DESC
BC8F05A0383F4948170003024DEA3C34	00000000	000000	000000	0000-00-00	0000-00-00	0.0	0.0		

2.2 Lire des champs spécifiques :

```
" step 2 - READ with Fields

  READ ENTITIES OF ZI_RAP_Travel_####

    ENTITY travel

      FIELDS ( AgencyID CustomerID )

    WITH VALUE #( ( TravelUUID = '<your uuid>' ) )

    RESULT DATA(travels).

  out->write( travels ).
```

```
15 CLASS zcl_rap_eml_1234 IMPLEMENTATION.
16   METHOD if_oo_adt_classrun~main.
17 *     " step 1 - READ
18 *     READ ENTITIES OF ZI_RAP_Travel_1234
19 *           ENTITY travel
20 *             FROM VALUE #( ( TravelUUID = 'BC8F05A0383F4948170003024DEA3C34' ) )
21 *           RESULT DATA(travels).
22 *
23 *     out->write( travels ).
24
25     " step 2 - READ with Fields
26     READ ENTITIES OF ZI_RAP_Travel_1234
27       ENTITY travel
28         FIELDS ( AgencyID CustomerID )
29       WITH VALUE #( ( TravelUUID = 'BC8F05A0383F4948170003024DEA3C34' ) )
30       RESULT DATA(travels).
31
32     out->write( travels ).
33
34   ENDMETHOD.
35
36 ENDCLASS.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'**application *console***.

L'opération **READ** est effectuée et le résultat s'affiche dans la console.
Les options **AgencyID** et **CustomerID** sont désormais également fournies dans le résultat.

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Console

ABAP Console

```
Table
TRAVELUUID                        TRAVELID  AGENCYID  CUSTOMERID  BEGINDATE   ENDDATE     BOOKINGFEE  TOTALPRICE  CURRENCYC
BC8F05A0383F4948170003024DEA3C34  00000000  000000    000000      0000-00-00  0000-00-00  0.0         0.0

Table
TRAVELUUID                        TRAVELID  AGENCYID  CUSTOMERID  BEGINDATE   ENDDATE     BOOKINGFEE  TOTALPRICE  CURRENCYC
BC8F05A0383F4948170003024DEA3C34  00000000  070042    000665      0000-00-00  0000-00-00  0.0         0.0
```

2.3 - Lire tous les champs :

```
" step 3 - READ with All Fields

  READ ENTITIES OF ZI_RAP_Travel_####

    ENTITY travel

      ALL FIELDS

    WITH VALUE #( ( TravelUUID = '<your uuid>' ) )

    RESULT DATA(travels).



  out->write( travels ).
```

```
25 *     " step 2 - READ with Fields
26 *     READ ENTITIES OF ZI_RAP_Travel_1234
27 *       ENTITY travel
28 *         FIELDS ( AgencyID CustomerID )
29 *       WITH VALUE #( ( TravelUUID = 'BC8F05A0383F4948170003024DEA3C34' ) )
30 *       RESULT DATA(travels).
31 *
32 *     out->write( travels ).
33
34     " step 3 - READ with All Fields
35      READ ENTITIES OF ZI_RAP_Travel_1234
36        ENTITY travel
37          ALL FIELDS
38        WITH VALUE #( ( TravelUUID = 'BC8F05A0383F4948170003024DEA3C34' ) )
39        RESULT DATA(travels).
40
41      out->write( travels ).
42
43    ENDMETHOD.
44
45  ENDCLASS.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'**application *console***.

L'opération **READ** est effectuée et le résultat s'affiche dans la console.
Tous les champs de l'entité de voyage sélectionnée sont désormais renvoyés.

```
Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros
Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Console
ABAP Console
TRAVELUUID                        TRAVELID  AGENCYID  CUSTOMERID  BEGINDATE   ENDDATE     BOOKINGFEE  TOTALPRICE  CURRENCYCODE  DESCRIPTION  TRAVELSTATUS  CREATEDBY  CRE
BC8F05A0383F4948170003024DEA3C34  00000000  000000    000000      0000-00-00  0000-00-00  0.0         0.0                                                          0.0

Table
TRAVELUUID                        TRAVELID  AGENCYID  CUSTOMERID  BEGINDATE   ENDDATE     BOOKINGFEE  TOTALPRICE  CURRENCYCODE  DESCRIPTION  TRAVELSTATUS  CREATEDBY  CRE
BC8F05A0383F4948170003024DEA3C34  00000000  070042    000665      0000-00-00  0000-00-00  0.0         0.0                                                          0.0

Table
TRAVELUUID                        TRAVELID  AGENCYID  CUSTOMERID  BEGINDATE   ENDDATE     BOOKINGFEE  TOTALPRICE  CURRENCYCODE  DESCRIPTION  TRAVELSTATUS  CREATEDBY  CRE
BC8F05A0383F4948170003024DEA3C34  00000004  070042    000665      2020-02-29  2020-12-27  40.0        1871.0      USD           Vacation     O             Rahn       202
```

2.4 - Lire via une association (_Booking) :

```
" step 4 - READ By Association

  READ ENTITIES OF ZI_RAP_Travel_####

    ENTITY travel BY \_Booking

      ALL FIELDS WITH VALUE #( ( TravelUUID = '<your uuid>' ) )

    RESULT DATA(bookings).

  out->write( bookings ).
```

```
34 *    " step 3 - READ with All Fields
35 *     READ ENTITIES OF ZI_RAP_Travel_1234
36 *       ENTITY travel
37 *         ALL FIELDS
38 *       WITH VALUE #( ( TravelUUID = 'BC8F05A0383F4948170003024DEA3C34' ) )
39 *       RESULT DATA(travels).
40 *
41 *     out->write( travels ).
42
43     " step 4 - READ By Association
44     READ ENTITIES OF ZI_RAP_Travel_1234
45       ENTITY travel BY \_Booking
46         ALL FIELDS WITH VALUE #( ( TravelUUID = 'BC8F05A0383F4948170003024DEA3C34' ) )
47       RESULT DATA(bookings).
48
49     out->write( bookings ).
50
51   ENDMETHOD.
52
53 ENDCLASS.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'**application *console***.

L'opération **READ** est effectuée et le résultat s'affiche dans la console.
Ainsi, toutes les entités de réservation associées à ce voyage sont récupérées avec tous leurs champs.

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Problems | Properties | Templates | Bookmarks | Transport Organizer | Feed Reader | Console

ABAP Console

Table

BOOKINGUUID	TRAVELUUID	BOOKINGID	BOOKINGDATE	CUSTOMERID	CARRIERID	CONNECTIONID	FLIGHTDATE	FLIGHTPRICE	CURRENCYCODE	CREATEDBY	LA
F78E0D3051CC464817000802A9F1BC79	BC8F05A0383F4948170003024DEA3C34	0001	2021-03-23	000509	AA	0322	2021-04-11	840.0	USD	Mustermar	
F88E0D3051CC464817000802A9F1BC79	BC8F05A0383F4948170003024DEA3C34	0002	2021-03-23	000301	AA	0322	2021-04-11	840.0	USD	Mustermar	
F98E0D3051CC464817000802A9F1BC79	BC8F05A0383F4948170003024DEA3C34	0003	2021-03-23	000649	AA	0322	2021-04-11	840.0	USD	Mustermar	

Étape 3 – Gestion des erreurs

Lire une entité inexistante pour déclencher un FAILED :

```
" step 5 - Unsuccessful READ

  READ ENTITIES OF ZI_RAP_Travel_####

    ENTITY travel

      ALL FIELDS WITH VALUE #( ( TravelUUID = '11111111111111111111111111111111' ) )

    RESULT DATA(travels)

    FAILED DATA(failed)
```

```
  REPORTED DATA(reported).


  out->write( travels ).

  out->write( failed ).    " complex structures not supported by the console output

  out->write( reported ).  " complex structures not supported by the console output
```

```
46  *        ALL FIELDS WITH VALUE #( ( TravelUUID = 'BC8F05A0383F4948170003024DEA3C34' ) )
47  *      RESULT DATA(bookings).
48  *
49  *    out->write( bookings ).
50
51     " step 5 - Unsuccessful READ
52     READ ENTITIES OF ZI_RAP_Travel_1234
53       ENTITY travel
54         ALL FIELDS WITH VALUE #( ( TravelUUID = '11111111111111111111111111111111' ) )
55       RESULT DATA(travels)
56       FAILED DATA(failed)
57       REPORTED DATA(reported).
58
59     out->write( travels ).
60     out->write( failed ).     " complex structures not supported by the console output
61     out->write( reported ).   " complex structures not supported by the console output
62
63
64   ENDMETHOD.
65
66 ENDCLASS.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'**application *console*** .

L'opération **READ** est exécutée et le résultat s'affiche dans la console.
Le tableau de résultats est vide. Les tables **failed** et **reported** sont des tables imbriquées qui ne sont pas prises en charge par la sortie de la console.

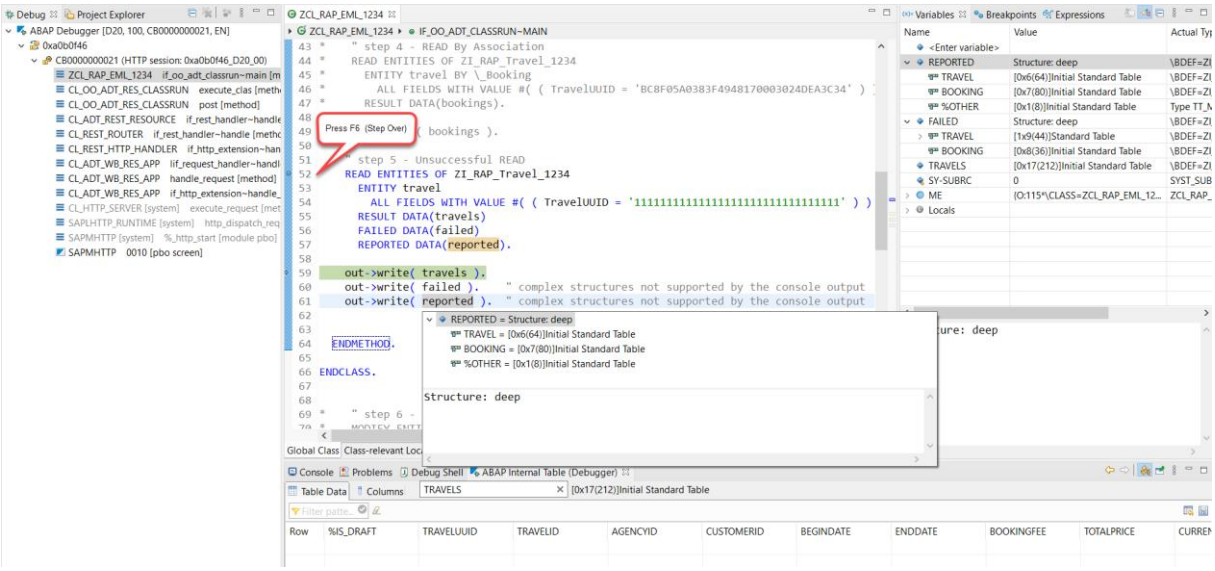
```
Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros
Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Console
ABAP Console
Table
TRAVELUUID  TRAVELID  AGENCYID  CUSTOMERID  BEGINDATE  ENDDATE  BOOKINGFEE  TOTALPRICE  CURRENCYCODE  DESCRIPTION  TRAVELSTATUS  CREATEDBY  CREATEDAT  LASTCHANGEDBY  LAST
Data type not yet supported ...
Data type not yet supported ...
```

Par conséquent, définissez un point d'arrêt sur la ligne d'instruction **READ ENTITIES**, exécutez (**F9**) la classe en tant que console d'application et vérifiez le contenu des tables dans le *débogueur ABAP* .

La table **Failed** Contient une entrée pour la *lecture* infructueuse avec la cause de l'échec **NOT_FOUND**.

La table **reported** est vide, car la cause de l'échec explique suffisamment la raison.

Débogueur ABAP :



Analyse des erreurs via le **debugger ABAP**, car FAILED/REPORTED ne s'affichent pas dans la console.

Etape 4 – Modification

```
REPORTED DATA(reported_commit). " step 6 - MODIFY Update

  MODIFY ENTITIES OF ZI_RAP_Travel_####

    ENTITY travel

      UPDATE

        SET FIELDS WITH VALUE

          #( ( TravelUUID  = '<your uuid>'

               Description = 'I like RAP@openSAP') )



  FAILED DATA(failed)
```

```
REPORTED DATA(reported).


out->write( 'Update done' ).
```

```
61  *    out->write( reported ).  " complex structures not supported by the console output
62
63     " step 6 - MODIFY Update
64     MODIFY ENTITIES OF ZI_RAP_Travel_1234
65       ENTITY travel
66         UPDATE
67           SET FIELDS WITH VALUE
68             #( ( TravelUUID  = 'BC8F05A0383F4948170003024DEA3C34'
69                  Description = 'I like RAP@openSAP' ) )
70
71      FAILED DATA(failed)
72      REPORTED DATA(reported).
73
74     out->write( 'Update done' ).
75
76   ENDMETHOD.
77
78 ENDCLASS.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'**application *console***.

L'opération **MODIFY** a été effectuée.
Vérifiez la *description* mise à jour dans l'aperçu *des données* de votre table *de voyage* **ZRAP_ATRAV_####** (**F8**) ou dans votre application *Voyage* .

Comme vous pouvez le constater **the description has not changed**, il s'agit toujours de l'ancienne valeur. Ceci est dû à l' instruction **missing commit** d'entités que nous allons ajouter.

Aperçu des données :

ZCL_RAP_EML_1234 | ZRAP_ATRAV_1234

Raw Data | Show Log | Max. Rows: 100

Filter patte... | 100 rows retrieved - 1 ms (partial result) | SQL Console | Number of Entries | Select Columns | Add filter

TRAVEL_UUID	TRAVEL_ID	AGENCY_ID	CUSTOME	TOTAL_PRICE	CURRENCY_CODE	DESCRIPTION	OVERALL_STA...	
B98F05A0383F4948170003024DEA3C34	00000001	070041	000594	1889.00	USD	Vacation	O	Pan
BA8F05A0383F4948170003024DEA3C34	00000002	070007	000608	900.00	USD	Business Trip for Christi...	O	Meie
BB8F05A0383F4948170003024DEA3C34	00000003	070046	000093	4164.00	USD	Vacation	O	Dete
BC8F05A0383F4948170003024DEA3C34	00000004	070042	000665	1871.00	USD	Vacation	O	Rahn
BD8F05A0383F4948170003024DEA3C34	00000005	070007	000225	992.00	USD	Business Trip for Kurt, I...	O	Sessl
BE8F05A0383F4948170003024DEA3C34	00000006	070049	000072	5586.00	USD	Vacation	O	Neut

Ajoutez l'instruction **COMMIT ENTITIES**.
Pour cela, copiez et insérez l'extrait de code ci-dessous après « le », **MODIFY** comme illustré

dans la capture d'écran ci-dessous.
N'oubliez pas de remplacer l'espace réservé ####** par le suffixe de votre choix.

```
" step 6b - Commit Entities

    COMMIT ENTITIES

      RESPONSE OF ZI_RAP_Travel_####

      FAILED     DATA(failed_commit)

      REPORTED   DATA(reported_commit).
```

Votre code source devrait ressembler à ceci :

```
63      " step 6 - MODIFY Update
64      MODIFY ENTITIES OF ZI_RAP_Travel_1234
65        ENTITY travel
66          UPDATE
67            SET FIELDS WITH VALUE
68              #( ( TravelUUID  = 'BC8F05A0383F4948170003024DEA3C34'
69                   Description = 'I like RAP@openSAP' ) )
70
71       FAILED DATA(failed)
72       REPORTED DATA(reported).
73
74      " step 6b - Commit Entities
75      COMMIT ENTITIES
76        RESPONSE OF ZI_RAP_Travel_1234
77        FAILED     DATA(failed_commit)
78        REPORTED   DATA(reported_commit).
79
80      out->write( 'Update done' ).
81
82    ENDMETHOD.
83
84  ENDCLASS.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'**application *console*** .

Consultez le résultat dans l'aperçu *des données* de votre table *de voyage* **ZRAP_ATRAV_####** (**F8**) ou dans votre application *Voyage* .

Vous pouvez filtrer les entrées par UUID *de voyage* (**TRAVEL_UUID** alias **TRAVELUUID**) ou par identifiant *de voyage* correspondant (TRAVEL_**ID**).

La *description* aurait dû être modifiée maintenant.

Aperçu des données :

ZCL_RAP_EML_1234 | ZRAP_ATRAV_1234

Raw Data | Show Log | Max. Rows: 100

Filter patte... | 100 rows retrieved - 12 ms (partial result) | SQL Console | Number of Entries | Select Columns | Add filter

C.	TRAVEL_UUID	TRAVEL_ID	AGENCY_ID	CUSTOMER_I	I_PRICE	CURRENCY_CODE	DESCRIPTION	OVERALL_STA...	C
	B98F05A0383F4948170003024DEA3C34	00000001	070041	000594	1889.00	USD	Vacation	O	Pan
	BA8F05A0383F4948170003024DEA3C34	00000002	070007	000608	900.00	USD	Business Trip for Christi...	O	Meie
	BB8F05A0383F4948170003024DEA3C34	00000003	070046	000093	4164.00	USD	Vacation	O	Dete
	BC8F05A0383F4948170003024DEA3C34	00000004	070042	000665	371.00	USD	I like RAP@openSAP	O	Rahn
	BD8F05A0383F4948170003024DEA3C34	00000005	070007	000225	992.00	USD	Business Trip for Kurt, I...	O	Sessl
	BE8F05A0383F4948170003024DEA3C34	00000006	070049	000072	586.00	USD	Vacation	O	Neub

---

Etape 5 – Création

Créea un nouveau Travel avec %cid :

```
" step 7 - MODIFY Create

  MODIFY ENTITIES OF ZI_RAP_Travel_####

    ENTITY travel

      CREATE

        SET FIELDS WITH VALUE

          #( ( %cid       = 'MyContentID_1'

               AgencyID    = '70012'

               CustomerID  = '14'

               BeginDate   = cl_abap_context_info=>get_system_date( )

               EndDate     = cl_abap_context_info=>get_system_date( ) + 10

               Description = 'I like RAP@openSAP' ) )


  MAPPED DATA(mapped)
```

```
 FAILED DATA(failed)

 REPORTED DATA(reported).


out->write( mapped-travel ).


COMMIT ENTITIES

  RESPONSE OF ZI_RAP_Travel_####

  FAILED     DATA(failed_commit)

  REPORTED   DATA(reported_commit).


out->write( 'Create done' ).
```

Ici, on récupère le **TravelUUID** généré via **mapped-travel**.

Votre code source devrait ressembler à ceci :

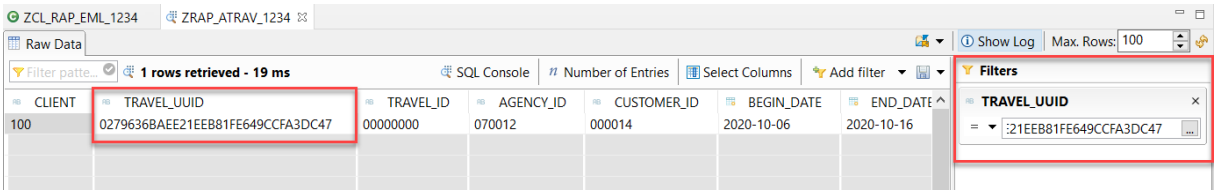
```
77  *      FAILED     DATA(failed_commit)
78  *      REPORTED   DATA(reported_commit).
79  *
80  *    out->write( 'Update done' ).
81
82     " step 7 - MODIFY Create
83     MODIFY ENTITIES OF ZI_RAP_Travel_1234
84       ENTITY travel
85         CREATE
86           SET FIELDS WITH VALUE
87             #( ( %cid        = 'MyContentID_1'
88                  AgencyID    = '70012'
89                  CustomerID  = '14'
90                  BeginDate   = cl_abap_context_info=>get_system_date( )
91                  EndDate     = cl_abap_context_info=>get_system_date( ) + 10
92                  Description = 'I like RAP@openSAP' ) )
93
94      MAPPED DATA(mapped)
95      FAILED DATA(failed)
96      REPORTED DATA(reported).
97
98     out->write( mapped-travel ).
99
100    COMMIT ENTITIES
101      RESPONSE OF ZI_RAP_Travel_1234
102      FAILED     DATA(failed_commit)
103      REPORTED   DATA(reported_commit).
104
105    out->write( 'Create done' ).
106
107  ENDMETHOD.
108
109 ENDCLASS.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'application *console* .

```
Problems  Properties  Templates  Bookmarks  Feed Reader  Transport Organizer  Console
ABAP Console
Update done

Table
%CID            TRAVELUUID
MyContentID_1   0279636BAEE21EEB81FE649CCFA3DC47
Create done
```

Vérifiez le résultat dans l'aperçu *des données* de votre table *de voyage* **ZRAP_ATRAV_####** (**F8**) ou dans votre application *Voyage* .
Vous pouvez filtrer les entrées par UUID *de voyage* (**TRAVEL_UUID**alias **TRAVELUUID**). Pour cela, vous pouvez utiliser l'UUID *de voyage* indiqué dans la *console* .

Aperçu des données :



Etape 6 – Suppression

Supprimer un Travel par UUID :

```
" step 8 - MODIFY Delete

  MODIFY ENTITIES OF ZI_RAP_Travel_####

    ENTITY travel

      DELETE FROM

        VALUE

          #( ( TravelUUID  = '<your uuid>' ) )


  FAILED DATA(failed)

  REPORTED DATA(reported).


 COMMIT ENTITIES

   RESPONSE OF ZI_RAP_Travel_####

   FAILED     DATA(failed_commit)

   REPORTED   DATA(reported_commit).
```

```
out->write( 'Delete done' ).
```

Votre code source devrait ressembler à ceci :

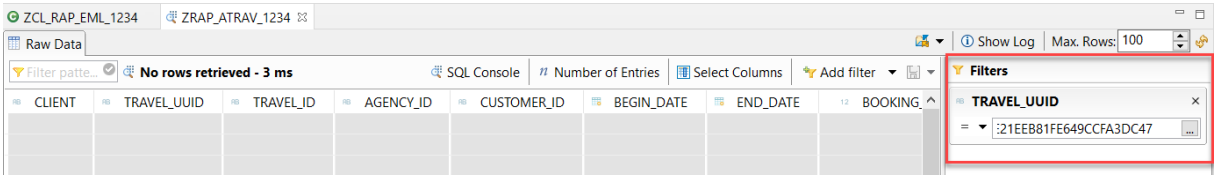
```
105 *    out->write( 'Create done' ).
106
107   " step 8 - MODIFY Delete
108     MODIFY ENTITIES OF ZI_RAP_Travel_1234
109       ENTITY travel
110         DELETE FROM
111           VALUE
112             #( ( TravelUUID  = '0279636BAEE21EEB81FE649CCFA3DC47' ) )
113
114      FAILED DATA(failed)
115      REPORTED DATA(reported).
116
117     COMMIT ENTITIES
118       RESPONSE OF ZI_RAP_Travel_1234
119       FAILED     DATA(failed_commit)
120       REPORTED   DATA(reported_commit).
121
122     out->write( 'Delete done' ).
123
124   ENDMETHOD.
125
126 ENDCLASS.
```

Enregistrez , activez et exécutez (**F9**) la classe en tant qu'**application *console***.

```
Problems  Properties  Templates  Bookmarks  Feed Reader  Transport Organizer  Console
ABAP Console
Table
%CID            TRAVELUUID
MyContentID_1   0279636BAEE21EEB81FE649CCFA3DC47
Create done

Delete done
```

Vérifiez le résultat dans l'aperçu *des données* de votre table *de voyage* **ZRAP_ATRAV_###** ou dans votre application *Voyage*.
Vous pouvez filtrer les entrées par UUID *de voyage* (TRAVEL**_UUID** alias **TRAVELUUID**).
Utilisez l'UUID indiqué dans la *console*.

Aperçu des données :



Résumé des opérations EML :

Opération	Syntaxe EML
Read	READ ENTITIES OF ...
Create	MODIFY ENTITIES OF ... CREATE
Update	MODIFY ENTITIES OF ... UPDATE
Delete	MODIFY ENTITIES OF ... DELETE FROM
Commit	COMMIT ENTITIES RESPONSE OF ...
Champs spécifiques	FIELDS (...) ou ALL FIELDS
Association	BY _AssociationName
Gestion erreurs	FAILED DATA(...), REPORTED DATA(...)

4.4 Amélioration du comportement des objets métier avec une logique spécifique à l'application (Unité 4)

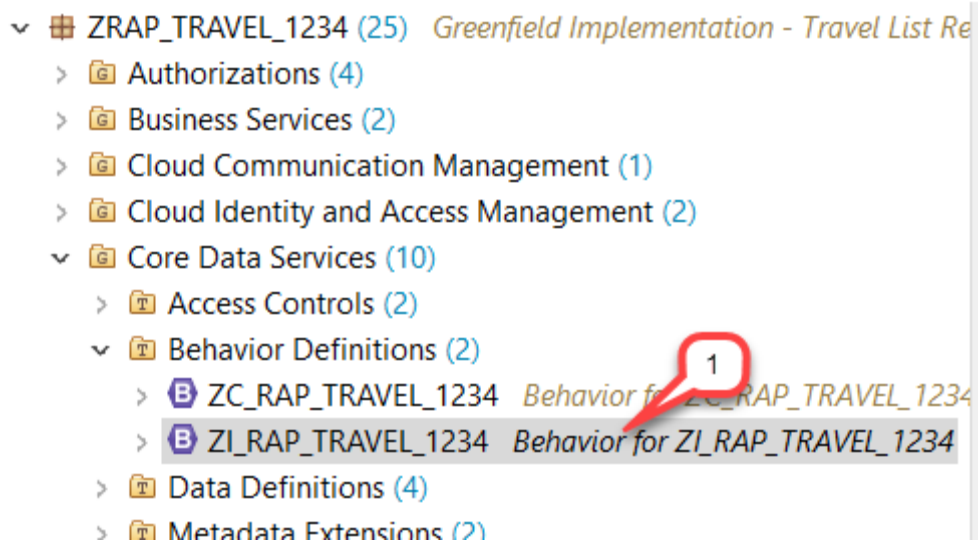
Objectif :

Apprendre à :

- Enrichir le **comportement d'un BO RAP** avec de la logique métier : **actions, déterminations, validations**
- Contrôler dynamiquement ou statiquement les champs (readonly, obligatoire)
- **Projeter les actions** pour les rendre visibles dans Fiori
- **Ajouter des boutons** dans l'UI grâce aux annotations dans l'extension CDS

Étapes clés de l'exercice :

Étape 1 – Améliorer la behavior definition du BO Travel



1.1 – Définir la classe de comportement :

```
implementation in class zbp_i_rap_travel_#### unique
```

1.2 – Définir des champs en lecture seule :

```
field ( readonly  ) TravelID, TotalPrice, TravelStatus;
```

```
field ( readonly  ) LastChangedAt, LastChangedBy, CreatedAt, CreatedBy, LocalLastChangedAt;
```

1.3 – Définir des champs obligatoires :

```
field ( mandatory ) AgencyID, CustomerID;
```

1.4 – Ajouter des actions :

```
action ( features : instance ) acceptTravel result [1] $self;

 action ( features : instance ) rejectTravel result [1] $self;

 internal action recalcTotalPrice;
```

1.5 – Ajouter des déterminations :

```
determination setInitialStatus    on modify { create; }

determination calculateTotalPrice on modify { field BookingFee, CurrencyCode; }

determination calculateTravelID   on save   { create; }
```

1.6 – Ajouter des validations :

```
validation validateAgency   on save { field AgencyID; create; }

 validation validateCustomer on save { field CustomerID; create; }

 validation validateDates    on save { field BeginDate, EndDate; create; }
```

Enregistrez la définition du comportement.

Étape 2 – Améliorer la behavior definition de l'entité Booking

2.1 – Définir la classe de comportement :

```
implementation in class zbp_i_rap_booking_#### unique
```

2.2 – Définir des champs readonly :

```
field ( readonly ) TravelUUID, BookingID;

 field ( readonly ) CreatedBy, LastChangedBy, LocalLastChangedAt;
```

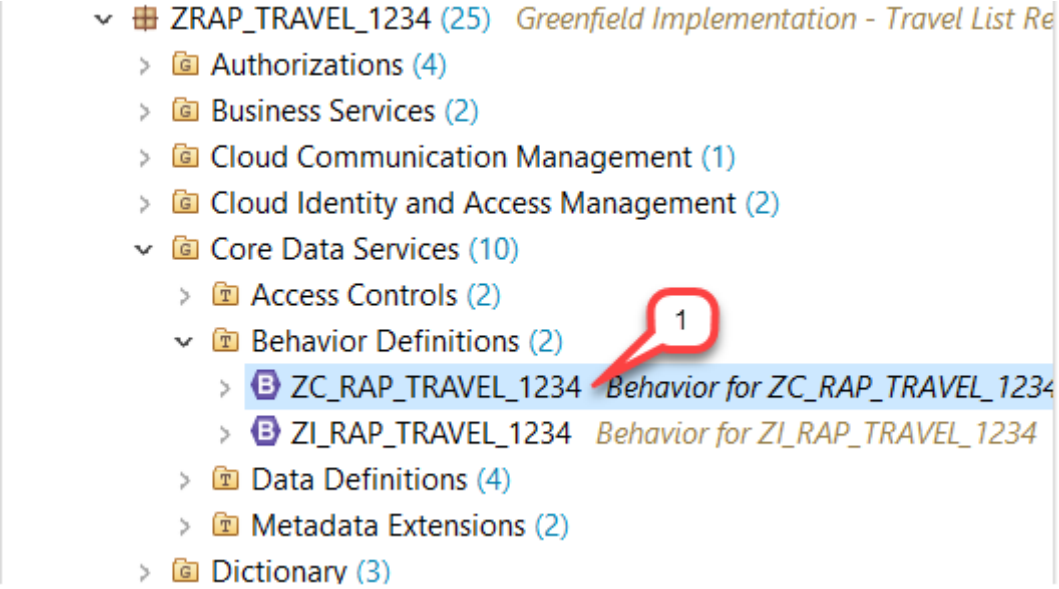
2.3 – Ajouter des déterminations :

```
determination calculateBookingID  on modify { create; }
```

```
determination calculateTotalPrice on modify { field FlightPrice, CurrencyCode; }
```

Enregistrez et activez la définition du comportement.

Étape 3 – Exposer les actions dans la behavior projection



Fichier : ZC_RAP_Travel_####

Ajouter les lignes suivantes :

```
use action acceptTravel;

 use action rejectTravel;
```

ZC_RAP_TRAVEL_1234

```
1  projection;
2
3  define behavior for ZC_RAP_Travel_1234 alias Travel
4  use etag
5  {
6    use create;
7    use update;
8    use delete;
9
10   use association _Booking { create; }
11
12   use action acceptTravel;
13   use action rejectTravel;
14 }
15
16 define behavior for ZC_RAP_Booking_1234 alias Booking
17 use etag
18 {
19   use update;
20   use delete;
21
22   use association _Travel;
23 }
```

Enregistrez et activez la définition du comportement.

Étape 4 – Ajouter les actions à l'interface Fiori

Fichier : ZC_RAP_Travel_#### (extension de métadonnées)

Modifier l'annotation @UI du champ TravelStatus :

```
@UI: { lineItem: [ { position: 90 },
                   { type: #FOR_ACTION, dataAction: 'acceptTravel', label: 'Accepter le voyage' },
                   { type: #FOR_ACTION, dataAction: 'rejectTravel', label: 'Rejeter le voyage' }
```

```
            ],
    identification: [ { position: 90 },
                { type: #FOR_ACTION, dataAction: 'acceptTravel', label: 'Accepter le voyage' },
                { type: #FOR_ACTION, dataAction: 'rejectTravel', label: 'Rejeter le voyage' }
            ] }
```

```
56  @UI: {  lineItem:       [ { position: 80 } ],
57          identification: [ { position: 80 } ] }
58  Description;
59
60  @UI: {  lineItem:       [ { position: 90 },
61                            { type: #FOR_ACTION, dataAction: 'acceptTravel', label: 'Accept Travel' },
62                            { type: #FOR_ACTION, dataAction: 'rejectTravel', label: 'Reject Travel' }
63                          ],
64          identification: [ { position: 90 },
65                            { type: #FOR_ACTION, dataAction: 'acceptTravel', label: 'Accept Travel' },
66                            { type: #FOR_ACTION, dataAction: 'rejectTravel', label: 'Reject Travel' }
67                          ] }
68  TravelStatus;
69
70  @UI.hidden: true
71  LastChangedAt;
```

Enregistrez et activez *l'extension CDS.*

Résumé final :

Élément	Description
Behavior Definition	Ajout d'actions, validations, déterminations
Contrôle statique	readonly, mandatory
Logic métier dynamique	Actions : acceptTravel, rejectTravel
Déterminations	setInitialStatus, calculateTotalPrice, calculateTravelID, calculateBookingID
Validations	Champs obligatoires et dates
Behavior Projection	Exposition explicite des actions

Élément	Description
UI CDS Extension	Ajout de boutons Fiori pour les actions via @UI annotations

4.5 MISE EN ŒUVRE DU COMPORTEMENT DE L'OBJET MÉTIER (Unité 5)

Objectif de l'unité

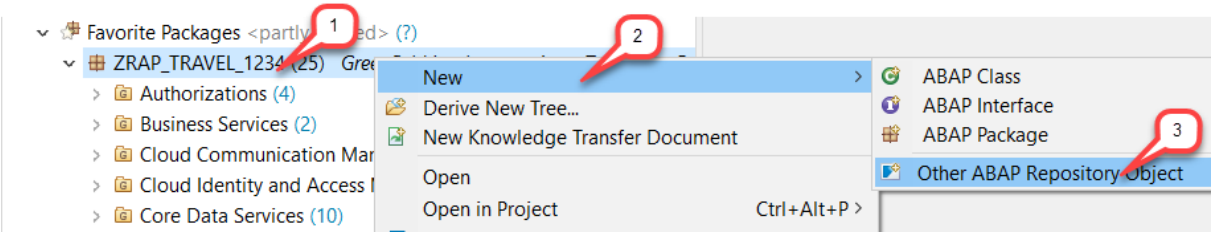
Dans cette unité, l'objectif est d'**implémenter la logique métier spécifique** dans le **pool de comportements** (classe de handler) de l'objet métier.
Tu vas apprendre à :

- Générer et structurer la classe d'implémentation de comportement (behavior pool) à partir de la définition de comportement.
- Ajouter des **constantes internes** liées à la logique métier (comme les statuts de l'objet Travel).
- Créer la classe de messages métier et la classe d'exception pour **gérer les erreurs de validation**.
- Préparer l'environnement pour écrire les méthodes métiers comme validate, check_before_save, finalize, etc.

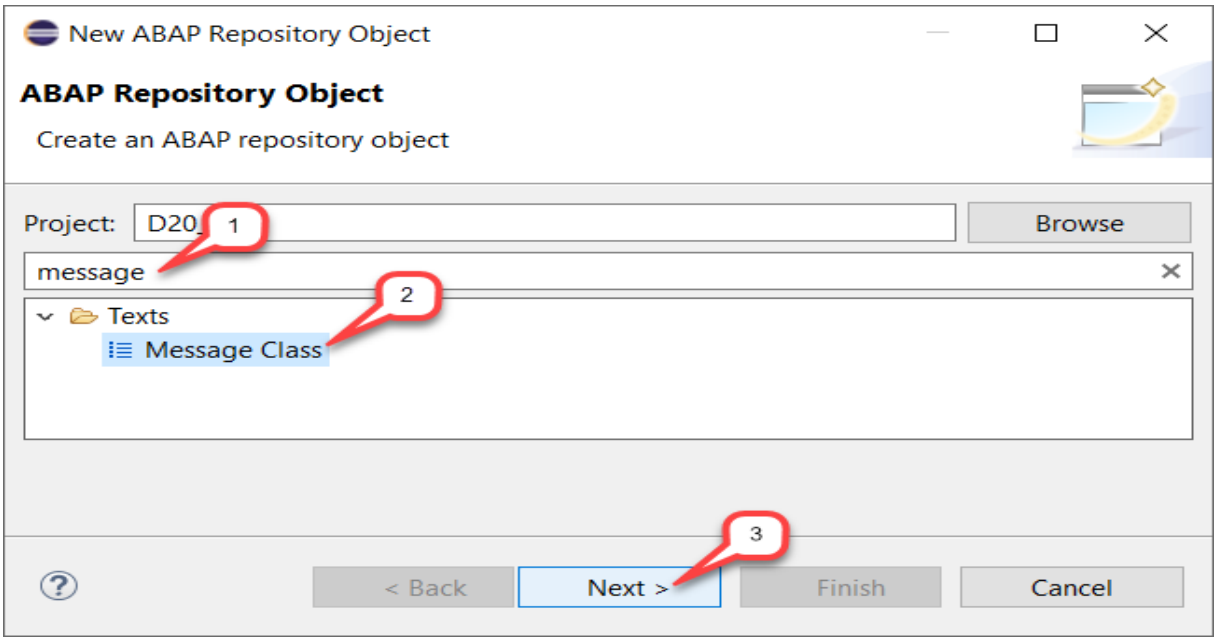
Étape 1 : Créer la classe de message ZRAP_MSG_####

Étapes détaillées

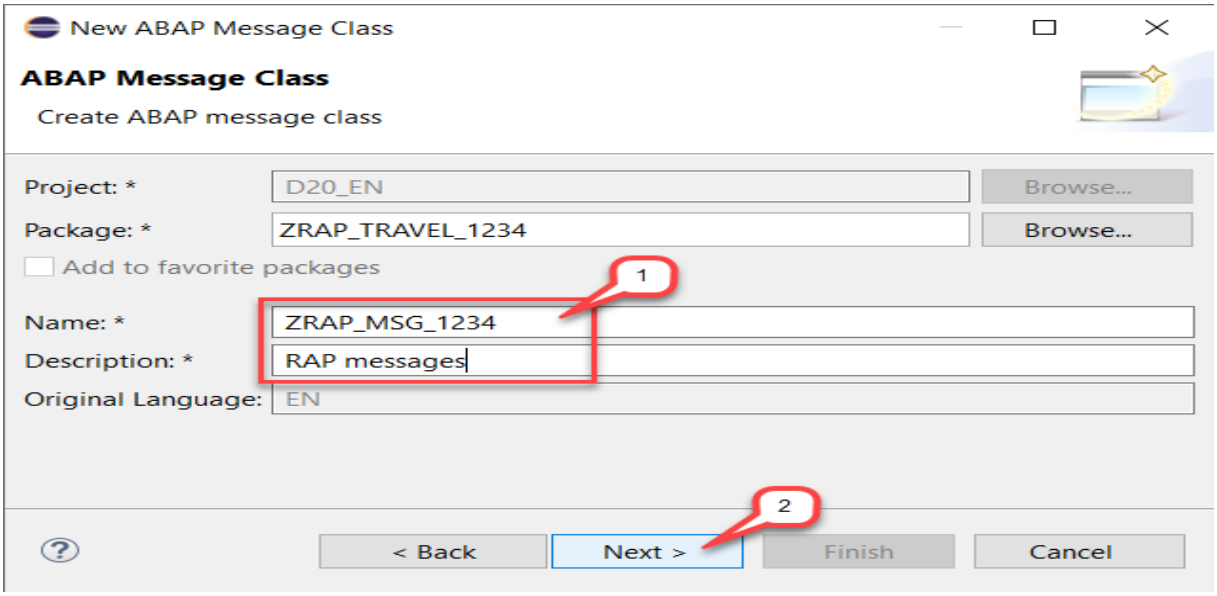
1. **Ouvrez votre package** dans l'ABAP Development Tools (ADT).
2. Clique droit sur le package → **New > Other ABAP Repository Object**.



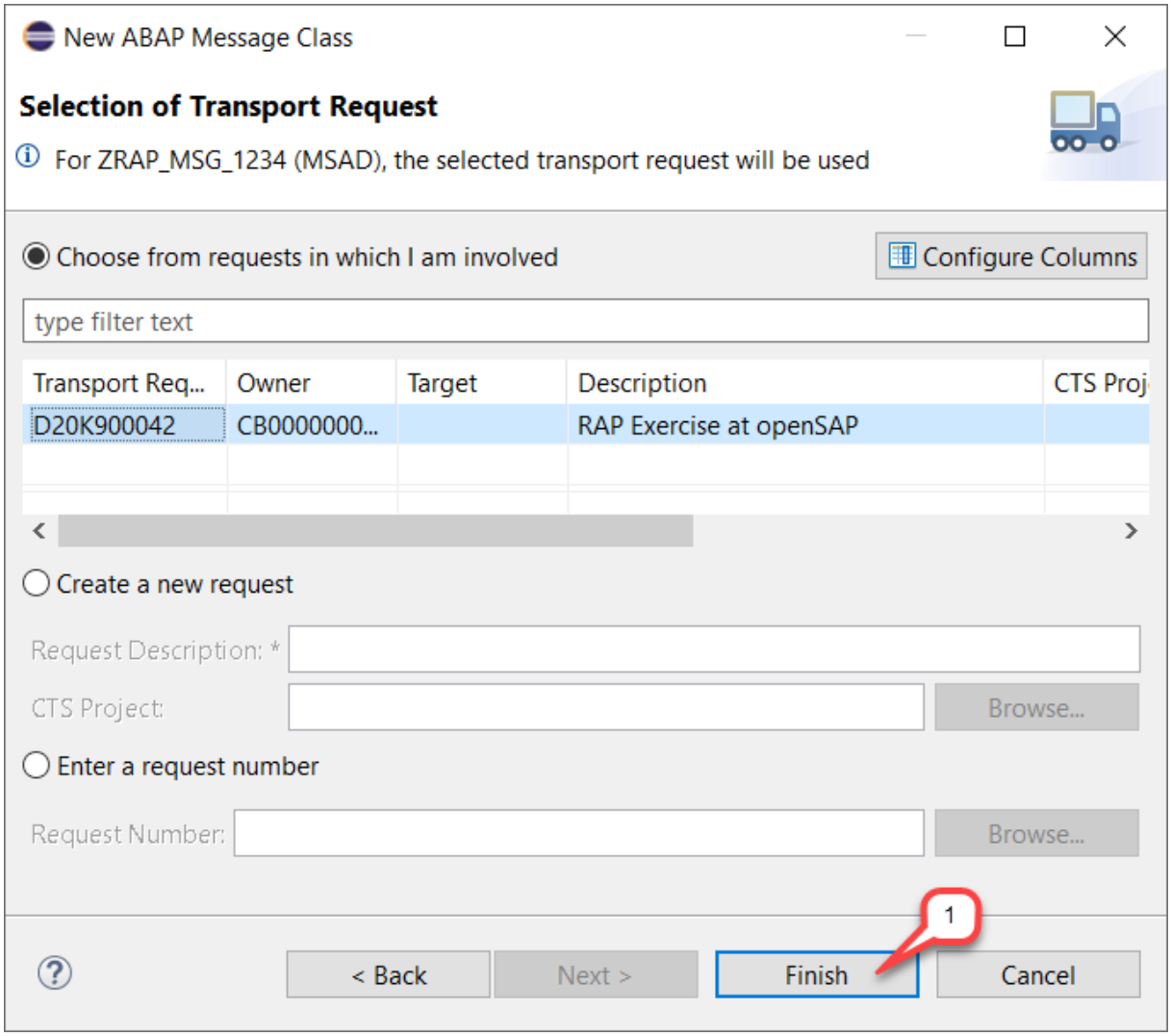
3. Dans l'assistant de création :
   - Tapez **message** dans la barre de filtre.
   - Sélectionnez **Message Class**, puis clique sur **Next**.



4. Renseignez les champs :
   - **Nom :** ZRAP_MSG_#### (remplace #### par votre suffixe personnel, max 4 caractères alphanumériques).
     👉 *Exemple :* ZRAP_MSG_1234
   - **Description :** Messages RAP (ou une autre description significative).
   - Cliquez sur **Next**.

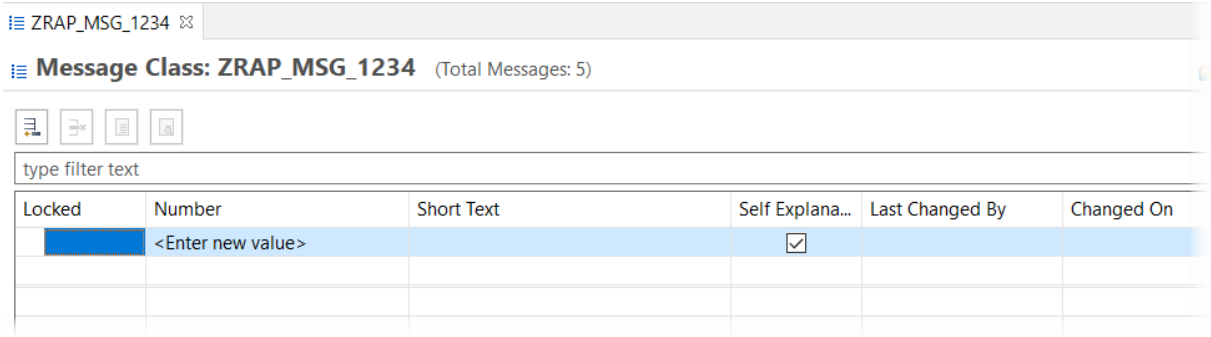


5. **Affectez une requête de transport** si elle est demandée.



6. Cliquez sur **Finish** pour générer l'objet.

7. L'éditeur s'ouvre automatiquement. Vous pouvez maintenant **ajouter les messages**.



Contenu de la classe de messages

Ajoutez les messages suivants dans la classe, de 001 à 005 :

ID	Message
001	Begin date &1 must not be after end date &2 for travel &3
002	Begin date &1 must be on or after system date
003	Customer &1 unknown
004	Agency &1 unknown
005	You are not authorized to perform this activity

Les **paramètres dynamiques** (&1, &2, &3) permettent d'insérer des valeurs variables dans le message lors de son appel.

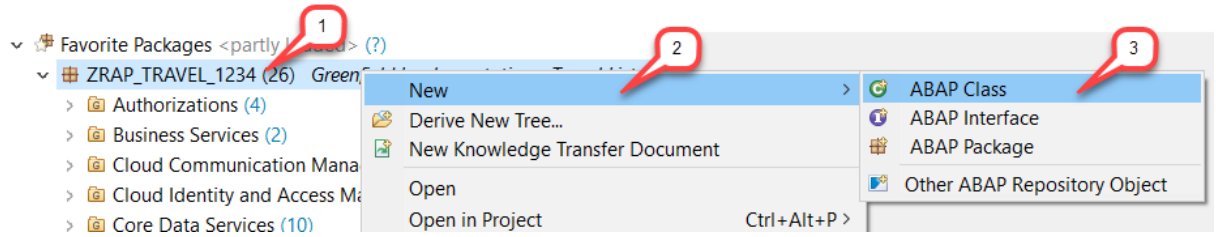
Enregistrez la nouvelle classe de message pour l'activer.

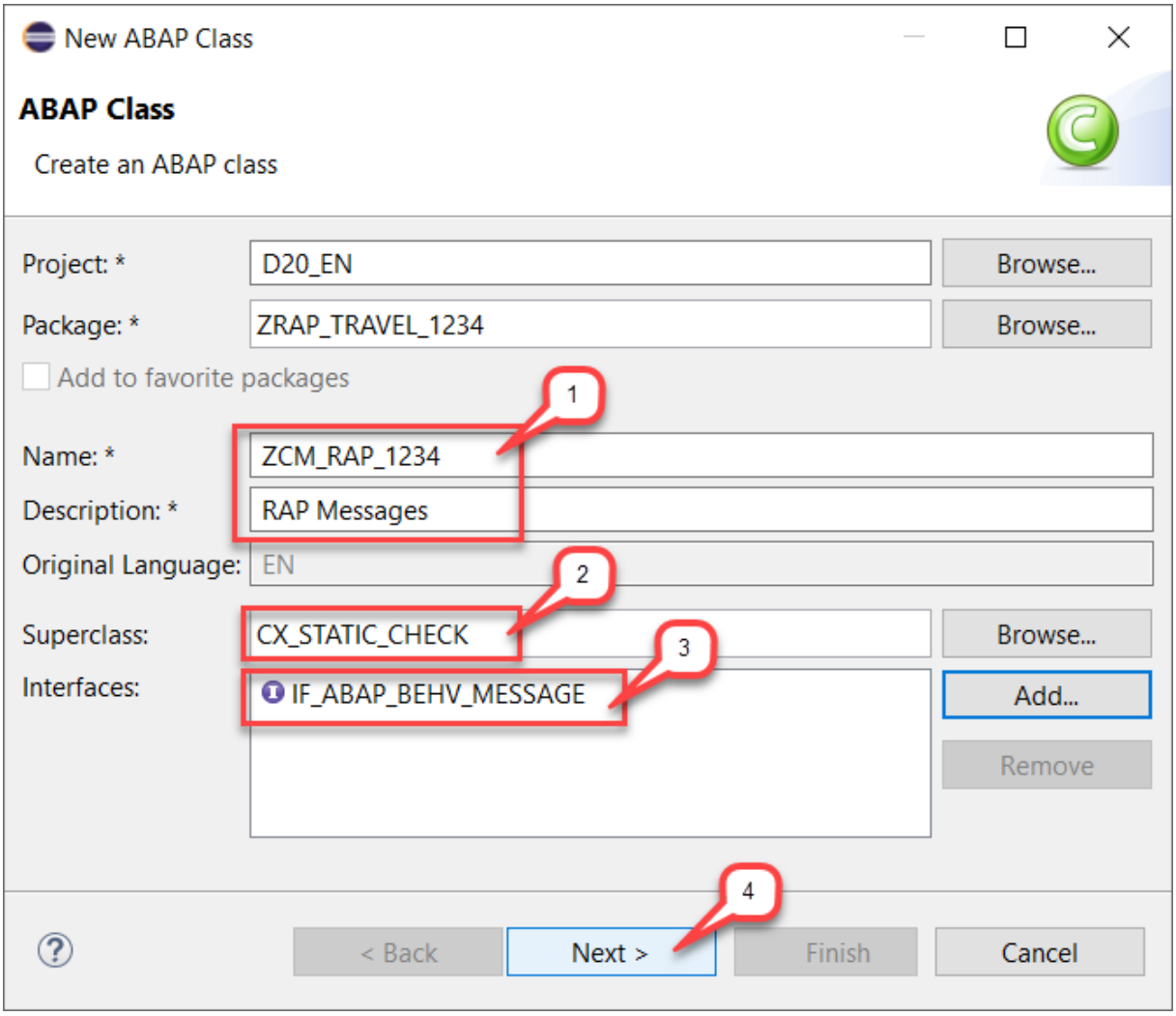
Étape 2 : Créer la classe d'exception ZCM_RAP_####

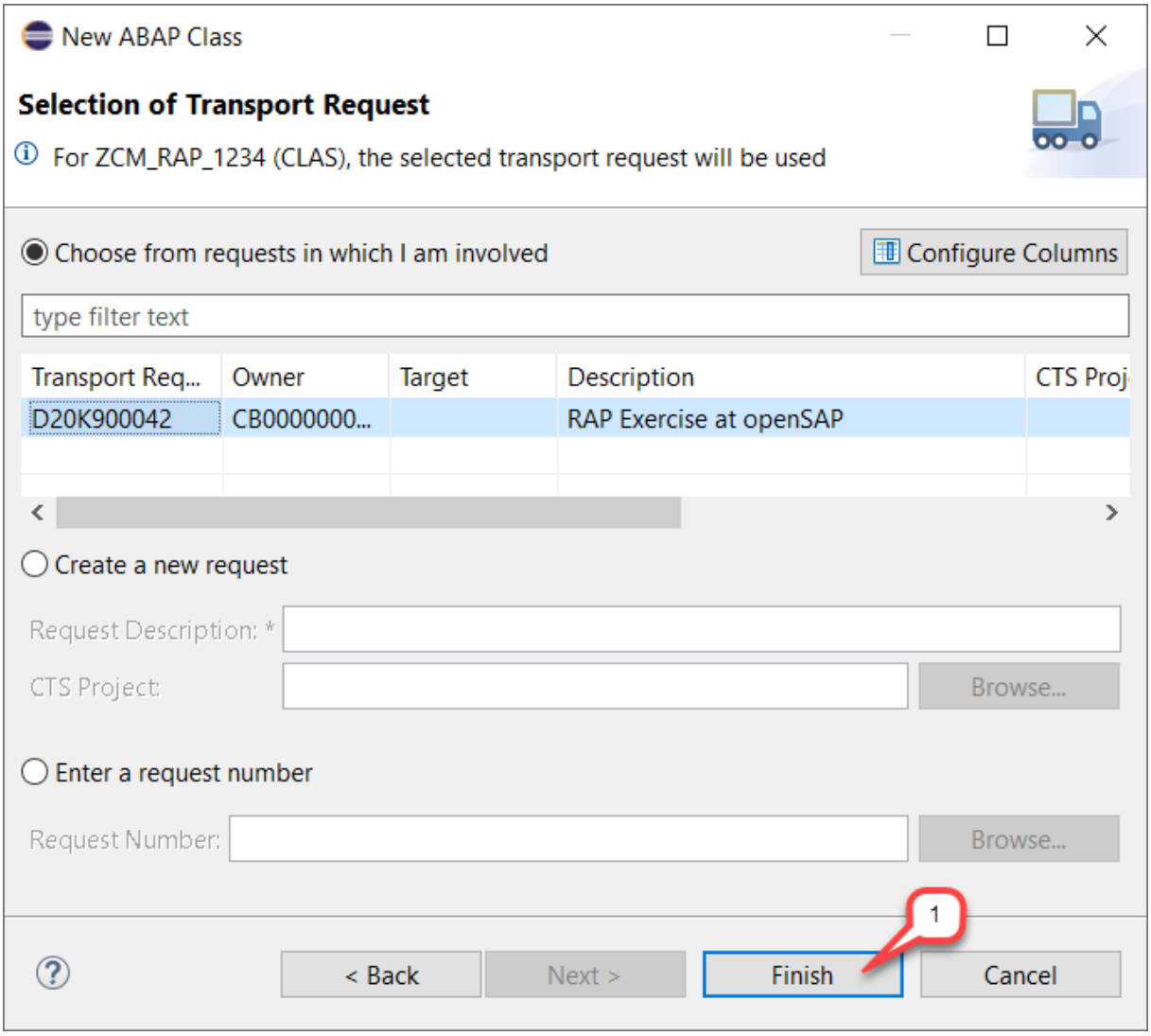
Étapes détaillées

1. Créer la classe ZCM_RAP_####

- Clic droit sur le package ZRAP_TRAVEL_#### → **New > ABAP Class**
- Nom : ZCM_RAP_####
- Description : *RAP Messages*
- Superclasse : CX_STATIC_CHECK
- Interface : IF_ABAP_BEHV_MESSAGE
- Suivant > Transport > Terminer


Favorite Packages
ZRAP_TRAVEL_1234 (26)
Authorizations (4)
Business Services (2)
Cloud Communication Mana
Cloud Identity and Access Ma
Core Data Services (10)
New
Derive New Tree...
New Knowledge Transfer Document
Open
Open in Project
Ctrl+Alt+P
ABAP Class
ABAP Interface
ABAP Package
Other ABAP Repository Object
1
2
3


New ABAP Class
ABAP Class
Create an ABAP class
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
Name: *
ZCM_RAP_1234
Description: *
RAP Messages
Original Language:
EN
Superclass:
CX_STATIC_CHECK
Browse...
Interfaces:
IF_ABAP_BEHV_MESSAGE
Add...
Remove
< Back
Next >
Finish
Cancel
1
2
3
4



---

2. Déclarer les constantes des messages (PUBLIC SECTION)

Dans la **section PUBLIC**, ajoutez les constantes représentant chaque message défini dans ZRAP_MSG_####.

```
CONSTANTS:

      BEGIN OF date_interval,

        msgid TYPE symsgid VALUE 'ZRAP_MSG_####',

        msgno TYPE symsgno VALUE '001',
```

```
      attr1 TYPE scx_attrname VALUE 'BEGINDATE',
      attr2 TYPE scx_attrname VALUE 'ENDDATE',
      attr3 TYPE scx_attrname VALUE 'TRAVELID',
      attr4 TYPE scx_attrname VALUE '',
    END OF date_interval .
  CONSTANTS:
    BEGIN OF begin_date_before_system_date,
      msgid TYPE symsgid VALUE 'ZRAP_MSG_####',
      msgno TYPE symsgno VALUE '002',
      attr1 TYPE scx_attrname VALUE 'BEGINDATE',
      attr2 TYPE scx_attrname VALUE '',
      attr3 TYPE scx_attrname VALUE '',
      attr4 TYPE scx_attrname VALUE '',
    END OF begin_date_before_system_date .
  CONSTANTS:
    BEGIN OF customer_unknown,
      msgid TYPE symsgid VALUE 'ZRAP_MSG_####',
      msgno TYPE symsgno VALUE '003',
      attr1 TYPE scx_attrname VALUE 'CUSTOMERID',
      attr2 TYPE scx_attrname VALUE '',
      attr3 TYPE scx_attrname VALUE '',
      attr4 TYPE scx_attrname VALUE '',
    END OF customer_unknown .
  CONSTANTS:
    BEGIN OF agency_unknown,
      msgid TYPE symsgid VALUE 'ZRAP_MSG_####',
```

```
      msgno TYPE symsgno VALUE '004',
      attr1 TYPE scx_attrname VALUE 'AGENCYID',
      attr2 TYPE scx_attrname VALUE '',
      attr3 TYPE scx_attrname VALUE '',
      attr4 TYPE scx_attrname VALUE '',
    END OF agency_unknown .
  CONSTANTS:
    BEGIN OF unauthorized,
      msgid TYPE symsgid VALUE 'ZRAP_MSG_####',
      msgno TYPE symsgno VALUE '005',
      attr1 TYPE scx_attrname VALUE '',
      attr2 TYPE scx_attrname VALUE '',
      attr3 TYPE scx_attrname VALUE '',
      attr4 TYPE scx_attrname VALUE '',
    END OF unauthorized .
```

ZCM_RAP_1234

ZCM_RAP_1234 ▸ DATE_INTERVAL

```
1  CLASS zcm_rap_1234 DEFINITION
2    PUBLIC
3    INHERITING FROM cx_static_check
4    FINAL
5    CREATE PUBLIC .
6
7    PUBLIC SECTION.
8
9      INTERFACES if_abap_behv_message .
10     INTERFACES if_t100_dyn_msg .
11     INTERFACES if_t100_message .
12
13     CONSTANTS:
14       BEGIN OF date_interval,
15         msgid TYPE symsgid VALUE 'ZRAP_MSG_1234',
16         msgno TYPE symsgno VALUE '001',
17         attr1 TYPE scx_attrname VALUE 'BEGINDATE',
18         attr2 TYPE scx_attrname VALUE 'ENDDATE',
19         attr3 TYPE scx_attrname VALUE 'TRAVELID',
20         attr4 TYPE scx_attrname VALUE '',
21       END OF date_interval .
22     CONSTANTS:
23       BEGIN OF begin_date_before_system_date,
24         msgid TYPE symsgid VALUE 'ZRAP_MSG_1234',
25         msgno TYPE symsgno VALUE '002',
26         attr1 TYPE scx_attrname VALUE 'BEGINDATE',
27         attr2 TYPE scx_attrname VALUE '',
28         attr3 TYPE scx_attrname VALUE '',
29         attr4 TYPE scx_attrname VALUE '',
30       END OF begin_date_before_system_date .
31     CONSTANTS:
32       BEGIN OF customer_unknown,
33         msgid TYPE symsgid VALUE 'ZRAP_MSG_1234',
34         msgno TYPE symsgno VALUE '003',
35         attr1 TYPE scx_attrname VALUE 'CUSTOMERID',
36         attr2 TYPE scx_attrname VALUE '',
37         attr3 TYPE scx_attrname VALUE '',
38         attr4 TYPE scx_attrname VALUE '',
39       END OF customer_unknown .
40     CONSTANTS:
41       BEGIN OF agency_unknown,
42         msgid TYPE symsgid VALUE 'ZRAP_MSG_1234',
43         msgno TYPE symsgno VALUE '004',
44         attr1 TYPE scx_attrname VALUE 'AGENCYID',
45         attr2 TYPE scx_attrname VALUE '',
46         attr3 TYPE scx_attrname VALUE '',
47         attr4 TYPE scx_attrname VALUE '',
48       END OF agency_unknown .
49     CONSTANTS:
50       BEGIN OF unauthorized,
51         msgid TYPE symsgid VALUE 'ZRAP_MSG_1234',
52         msgno TYPE symsgno VALUE '005',
53         attr1 TYPE scx_attrname VALUE '',
54         attr2 TYPE scx_attrname VALUE '',
55         attr3 TYPE scx_attrname VALUE '',
56         attr4 TYPE scx_attrname VALUE '',
57       END OF unauthorized .
58
59     METHODS constructor
60       IMPORTING
```

3. Redéfinir la méthode constructor

Supprimer la déclaration par défaut et insérer :

```
METHODS constructor

      IMPORTING

        severity   TYPE if_abap_behv_message=>t_severity DEFAULT
if_abap_behv_message=>severity-error

        textid     LIKE if_t100_message=>t100key OPTIONAL

        previous   TYPE REF TO cx_root OPTIONAL

        begindate  TYPE /dmo/begin_date OPTIONAL

        enddate    TYPE /dmo/end_date OPTIONAL

        travelid   TYPE /dmo/travel_id OPTIONAL

        customerid TYPE /dmo/customer_id OPTIONAL

        agencyid   TYPE /dmo/agency_id  OPTIONAL.
```

```
        attr4 TYPE scx_attrname VALUE '',
      END OF unauthorized .

    METHODS constructor
      IMPORTING
        severity   TYPE if_abap_behv_message=>t_severity DEFAULT if_abap_behv_message=>severity-error
        textid     LIKE if_t100_message=>t100key OPTIONAL
        previous   TYPE REF TO cx_root OPTIONAL
        begindate  TYPE /dmo/begin_date OPTIONAL
        enddate    TYPE /dmo/end_date OPTIONAL
        travelid   TYPE /dmo/travel_id OPTIONAL
        customerid TYPE /dmo/customer_id OPTIONAL
        agencyid   TYPE /dmo/agency_id  OPTIONAL.

  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
```

4. Définir les attributs nécessaires

Dans la section PRIVATE, ajouter :

```
DATA begindate TYPE /dmo/begin_date READ-ONLY.

    DATA enddate TYPE /dmo/end_date READ-ONLY.
```

```
    DATA travelid TYPE string READ-ONLY.

    DATA customerid TYPE string READ-ONLY.

    DATA agencyid TYPE string READ-ONLY.
```

```
59    METHODS constructor
60      IMPORTING
61        severity   TYPE if_abap_behv_message=>t_severity DEFAULT if_abap_behv_message=>severity-error
62        textid     LIKE if_t100_message=>t100key OPTIONAL
63        previous   TYPE REF TO cx_root OPTIONAL
64        begindate  TYPE /dmo/begin_date OPTIONAL
65        enddate    TYPE /dmo/end_date OPTIONAL
66        travelid   TYPE /dmo/travel_id OPTIONAL
67        customerid TYPE /dmo/customer_id OPTIONAL
68        agencyid   TYPE /dmo/agency_id  OPTIONAL.
69
70    DATA begindate TYPE /dmo/begin_date READ-ONLY.
71    DATA enddate TYPE /dmo/end_date READ-ONLY.
72    DATA travelid TYPE string READ-ONLY.
73    DATA customerid TYPE string READ-ONLY.
74    DATA agencyid TYPE string READ-ONLY.
75
76  PROTECTED SECTION.
77  PRIVATE SECTION.
```

---

5. Implémenter la méthode constructor

Dans l'implémentation de la méthode, ajouter :

```
me->if_abap_behv_message~m_severity = severity.

    me->begindate = begindate.

    me->enddate = enddate.

    me->travelid = |{ travelid ALPHA = OUT }|.

    me->customerid = |{ customerid ALPHA = OUT }|.

    me->agencyid = |{ agencyid ALPHA = OUT }|.
```

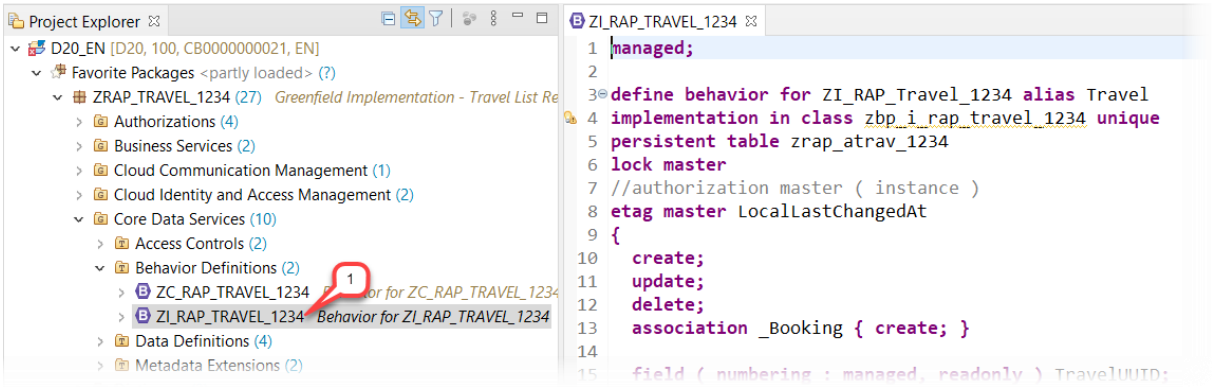
```
CLASS zcm_rap_1234 IMPLEMENTATION.

  METHOD constructor ##ADT_SUPPRESS_GENERATION.
    CALL METHOD super->constructor
      EXPORTING
        previous = previous.
    CLEAR me->textid.
    IF textid IS INITIAL.
      if_t100_message~t100key = if_t100_message=>default_textid.
    ELSE.
      if_t100_message~t100key = textid.
    ENDIF.

    me->if_abap_behv_message~m_severity = severity.
    me->begindate = begindate.
    me->enddate = enddate.
    me->travelid = |{ travelid ALPHA = OUT }|.
    me->customerid = |{ customerid ALPHA = OUT }|.
    me->agencyid = |{ agencyid ALPHA = OUT }|.

  ENDMETHOD.
ENDCLASS.
```

*Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

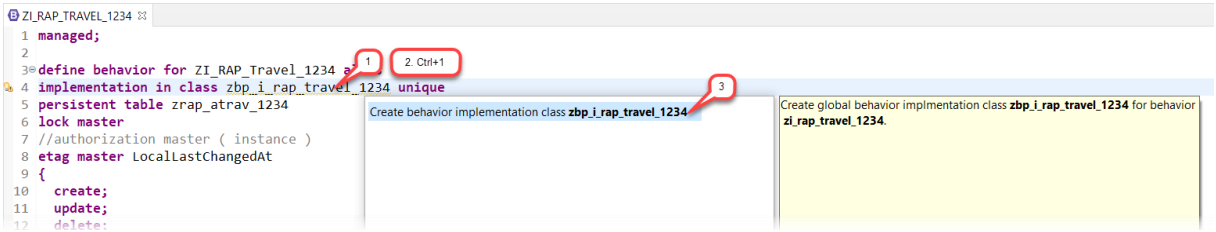
Enregistrez et activez la classe ABAP.

Étape 3 : Implémenter le pool de comportements pour l'entité *Voyage*

1. Ouvrez la *définition du comportement* **ZI_RAP_Travel_####** de base de votre objet métier dans l'Explorateur *de projets* .4.5

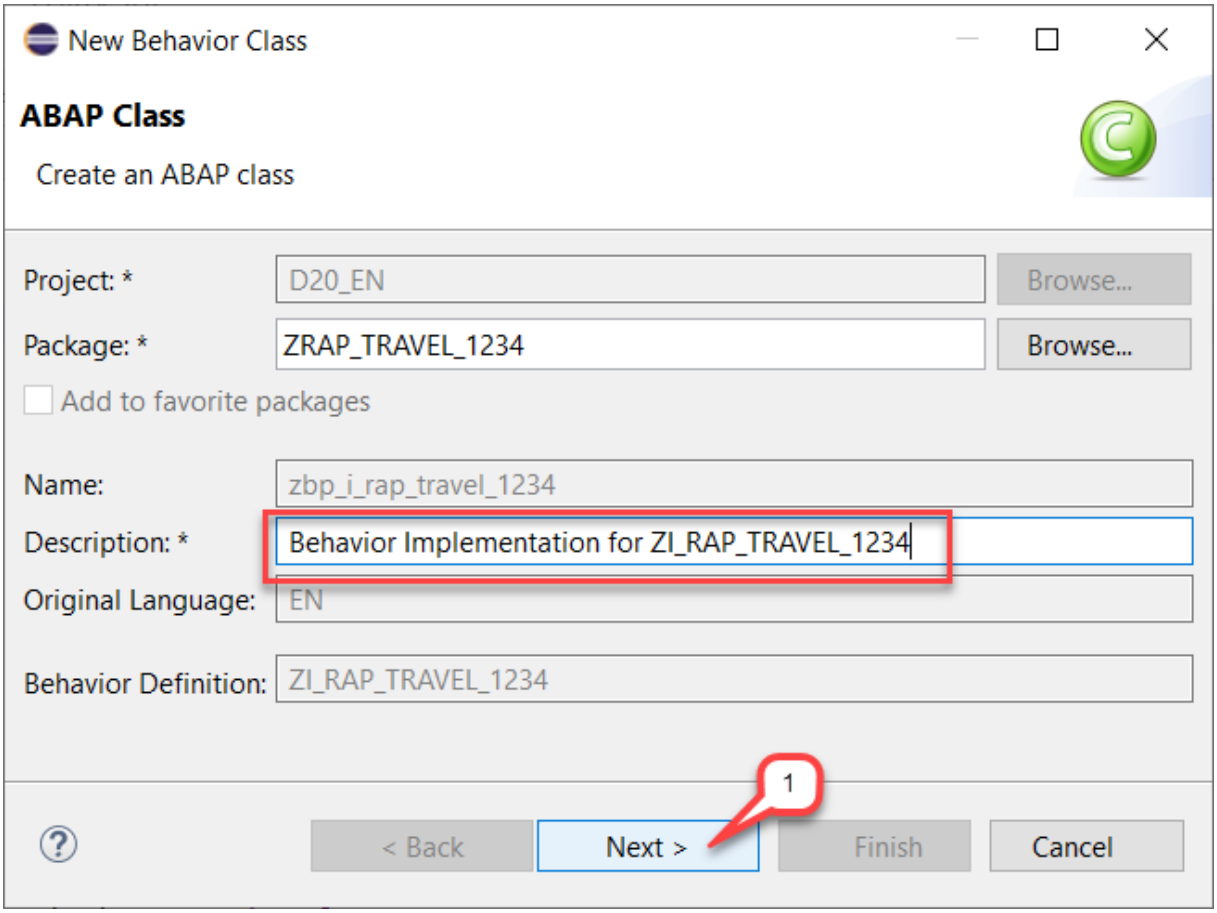


2. Placez le curseur sur le nom de la classe d'implémentation du comportement **zbp_i_rap_travel_####**,
démarrez la boîte de dialogue de Correction rapide à l'aide du raccourci **Ctrl+1** et choisissez **create…** pour générer l'implémentation du comportement pour l'entité Voyage.

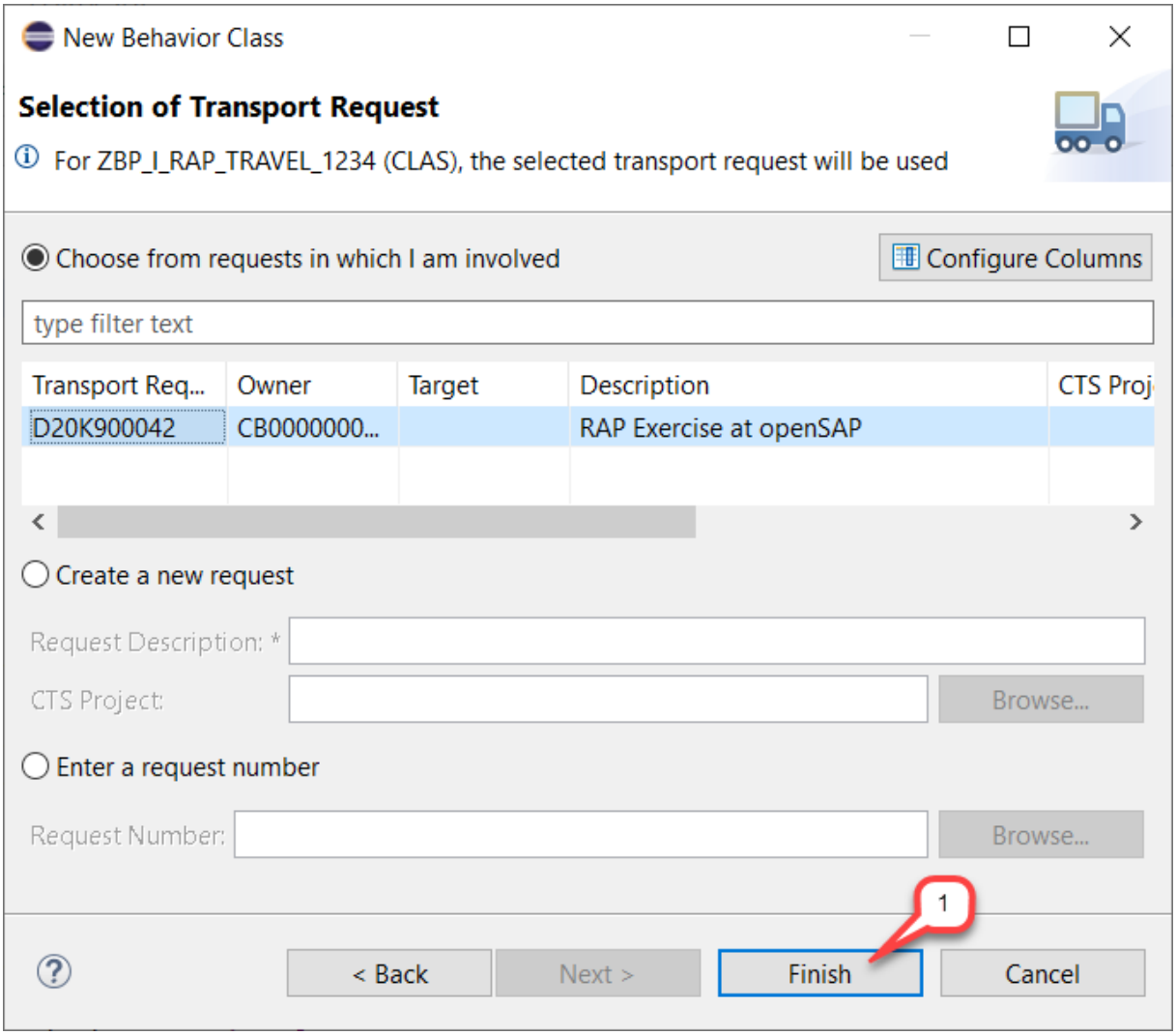


3. Dans l'assistant qui apparaît, conservez les champs par défaut et choisissez **Suivant >** .

   Le projet , *le package* et *la définition du comportement* ont été automatiquement attribués.



4. Attribuez une demande de transport et choisissez **Terminer**.



Le squelette de l'implémentation du comportement est généré et affiché dans l'éditeur.

Onglets ***Classe globale*** et ***Types locaux :***

ZI_RAP_TRAVEL_1234 | ZBP_I_RAP_TRAVEL_1234

```
CLASS zbp_i_rap_travel_1234 DEFINITION PUBLIC ABSTRACT FINAL FOR BEHAVIOR OF zi_rap_travel_1234.
ENDCLASS.

CLASS zbp_i_rap_travel_1234 IMPLEMENTATION.
ENDCLASS.
```

Global Class | Class-relevant Local Types | Local Types | Test Classes

ZI_RAP_TRAVEL_1234 | ZBP_I_RAP_TRAVEL_1234 | LHC_TRAVEL

```
CLASS lhc_Travel DEFINITION INHERITING FROM cl_abap_behavior_handler.
  PRIVATE SECTION.

    METHODS calculateTotalPrice FOR DETERMINE ON MODIFY
      IMPORTING keys FOR Travel~calculateTotalPrice.

    METHODS calculateTravelID FOR DETERMINE ON SAVE
      IMPORTING keys FOR Travel~calculateTravelID.

    METHODS setInitialStatus FOR DETERMINE ON MODIFY
      IMPORTING keys FOR Travel~setInitialStatus.

    METHODS validateAgency FOR VALIDATE ON SAVE
      IMPORTING keys FOR Travel~validateAgency.

    METHODS validateCustomer FOR VALIDATE ON SAVE
      IMPORTING keys FOR Travel~validateCustomer.

    METHODS validateDates FOR VALIDATE ON SAVE
      IMPORTING keys FOR Travel~validateDates.

    METHODS acceptTravel FOR MODIFY
      IMPORTING keys FOR ACTION Travel~acceptTravel RESULT result.

    METHODS rejectTravel FOR MODIFY
      IMPORTING keys FOR ACTION Travel~rejectTravel RESULT result.

    METHODS recalcTotalPrice FOR MODIFY
      IMPORTING keys FOR ACTION Travel~recalcTotalPrice.

    METHODS get_features FOR FEATURES
      IMPORTING keys REQUEST requested_features FOR Travel RESULT result.

ENDCLASS.

CLASS lhc_Travel IMPLEMENTATION.

  METHOD calculateTotalPrice.
  ENDMETHOD.

  METHOD calculateTravelID.
  ENDMETHOD.

  METHOD setInitialStatus.
  ENDMETHOD.

  METHOD validateAgency.
```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Dans l'onglet ***Classe globale*** , vous pouvez voir l'ajout **FOR BEHAVIOR OF** indiquant que cette classe fournit l'implémentation du comportement de l'objet métier spécifié (l'entité de voyage dans le cas présent).
L'implémentation appropriée s'effectue dans l'onglet ***Types locaux*** , où l'assistant a généré la classe de gestionnaire local **lhc_handler** héritant de cl_abap_behavior_handler.

5. Dans l'onglet ***Types locaux*** , ajoutez les constantes pour le statut de voyage dans le **PRIVATE SECTION**:

```
CONSTANTS:

      BEGIN OF travel_status,

        open      TYPE c LENGTH 1  VALUE 'O', " Open
```

```
      accepted TYPE c LENGTH 1  VALUE 'A', " Accepted

      canceled TYPE c LENGTH 1  VALUE 'X', " Cancelled

    END OF travel_status.
```

Votre code source devrait ressembler à ceci :

```
ZBP_I_RAP_TRAVEL_1234
ZBP_I_RAP_TRAVEL_1234 ▸ LHC_TRAVEL ▸ TRAVEL_STATUS
1  CLASS lhc_Travel DEFINITION INHERITING FROM cl_abap_behavior_handler.
2    PRIVATE SECTION.
3
4      CONSTANTS:
5        BEGIN OF travel_status,
6          open     TYPE c LENGTH 1  VALUE 'O', " Open
7          accepted TYPE c LENGTH 1  VALUE 'A', " Accepted
8          canceled TYPE c LENGTH 1  VALUE 'X', " Cancelled
9        END OF travel_status.
10
11     METHODS calculateTotalPrice FOR DETERMINE ON MODIFY
12       IMPORTING keys FOR Travel~calculateTotalPrice.
13
14     METHODS calculateTravelID FOR DETERMINE ON SAVE
```

6. Maintenant, implémentez la méthode **acceptTravel** qui modifie le statut global des instances *de voyage en accepté*.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **acceptTravel**.

```
 " Set the new overall status

   MODIFY ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

     ENTITY Travel

        UPDATE

          FIELDS ( TravelStatus )

          WITH VALUE #( FOR key IN keys

                          ( %tky          = key-%tky

                            TravelStatus = travel_status-accepted ) )

     FAILED failed
```

```
      REPORTED reported.
```

Veuillez noter :
l'ajout **IN LOCAL MODE** nous permet de manipuler des champs (par exemple, de modifier les champs en lecture seule) tout en ignorant le contrôle des fonctionnalités et des autorisations.

Après avoir modifié l'état *du trajet*, l'action doit fournir le résultat défini dans la définition du comportement. Dans notre cas, le résultat est **$self**, ce qui signifie que vous devez renvoyer l'instance de trajet avec toutes les valeurs des clés données.

```
" Fill the response table

    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel

        ALL FIELDS WITH CORRESPONDING #( keys )

      RESULT DATA(travels).


    result = VALUE #( FOR travel IN travels

                        ( %tky   = travel-%tky

                          %param = travel ) ).
```

Excursus sur l'utilisation de %tky:
%tky signifie la *clé transactionnelle*.

- Dans les cas d'utilisation non brouillons – comme celui-ci –, **%tky** contient la même valeur que **%key** celle qui est la clé de l'entité associée.
- Dans les cas d'utilisation avec brouillon activé, l'indicateur *is_draft* **%tky** sera automatiquement inclus. Vous gérerez l'activation du brouillon dans l'unité suivante.

Par conséquent, l'utilisation **%tky** dans un cas d'utilisation non brouillon réduit la nécessité de retravailler votre implémentation lorsque vous activez la gestion des brouillons dans votre définition de comportement, car le code peut gérer les instances actives ainsi que les brouillons.

Votre code source devrait ressembler à ceci :

```
62
63  METHOD acceptTravel.
64    " Set the new overall status
65    MODIFY ENTITIES OF zi_rap_travel_1234 IN LOCAL MODE
66      ENTITY Travel
67         UPDATE
68           FIELDS ( TravelStatus )
69           WITH VALUE #( FOR key IN keys
70                          ( %tky         = key-%tky
71                            TravelStatus = travel_status-accepted ) )
72      FAILED failed
73      REPORTED reported.
74
75    " Fill the response table
76    READ ENTITIES OF zi_rap_travel_1234 IN LOCAL MODE
77      ENTITY Travel
78        ALL FIELDS WITH CORRESPONDING #( keys )
79      RESULT DATA(travels).
80
81    result = VALUE #( FOR travel IN travels
82                      ( %tky   = travel-%tky
83                        %param = travel ) ).
84  ENDMETHOD.
```

7. L'implémentation de l'action **rejectTravel** est très similaire à celle de l'action **acceptTravel**. Elle modifie le statut du voyage en *« rejeté »*.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **rejectTravel**.

```
" Set the new overall status

    MODIFY ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel

         UPDATE

           FIELDS ( TravelStatus )

           WITH VALUE #( FOR key IN keys

                          ( %tky         = key-%tky

                            TravelStatus = travel_status-canceled ) )

      FAILED failed

      REPORTED reported.
```

```
    " Fill the response table
    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE
      ENTITY Travel
        ALL FIELDS WITH CORRESPONDING #( keys )
      RESULT DATA(travels).

    result = VALUE #( FOR travel IN travels
                        ( %tky   = travel-%tky
                          %param = travel ) ).
```

8. Vous allez maintenant implémenter la validation **validateAgency** qui vérifie les éléments fournis **AgencyID** lors de *la sauvegarde* – lorsqu'ils sont modifiés dans une instance existante ou qu'une nouvelle instance est créée.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **validateAgency**.

Les implémentations de validation commencent généralement par la lecture des données requises à l'aide d'EML.

Dans le cas présent, vous devez lire les informations **AgencyID** relatives aux clés fournies

```
" Read relevant travel instance data
    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE
      ENTITY Travel
        FIELDS ( AgencyID ) WITH CORRESPONDING #( keys )
      RESULT DATA(travels).
```

Ensuite, dérivez une table interne avec tous les éléments distincts **AgencyID** et effectuez une *sélection de base de données* pour valider l'existence.

```
DATA agencies TYPE SORTED TABLE OF /dmo/agency WITH UNIQUE KEY agency_id.


    " Optimization of DB select: extract distinct non-initial agency IDs
```

```
    agencies = CORRESPONDING #( travels DISCARDING DUPLICATES MAPPING agency_id =
AgencyID EXCEPT * ).

    DELETE agencies WHERE agency_id IS INITIAL.



    IF agencies IS NOT INITIAL.

      " Check if agency ID exist

      SELECT FROM /dmo/agency FIELDS agency_id

        FOR ALL ENTRIES IN @agencies

        WHERE agency_id = @agencies-agency_id

        INTO TABLE @DATA(agencies_db).

    ENDIF.
```

Et enfin, parcourez la table de déplacement, vérifiez si un **AgencyID** a été fourni et s'il existe, et émettez des messages d'état si un invalide **AgencyID** a été fourni.

Si l'élément est vide ou absent de la table de contrôle, un message est généré via notre classe d'exception. Lorsque vous utilisez les messages d'état, vous devez d'abord effacer les messages existants pour la *zone d'état* **AgencyID** de cette instance, puis en générer de nouveaux, si nécessaire.

Remarque :
les messages d'état sont importants dans le contexte d'un *projet* , où ils sont stockés avec l'état de l'instance. Dans un cas d'utilisation hors projet, les messages d'état sont traduits en messages transitoires par le framework.

```
" Raise msg for non existing and initial agencyID

    LOOP AT travels INTO DATA(travel).

      " Clear state messages that might exist

      APPEND VALUE #(  %tky               = travel-%tky

                       %state_area        = 'VALIDATE_AGENCY' )

        TO reported-travel.
```

```
      IF travel-AgencyID IS INITIAL OR NOT line_exists( agencies_db[ agency_id =
travel-AgencyID ] ).

        APPEND VALUE #( %tky = travel-%tky ) TO failed-travel.



        APPEND VALUE #( %tky        = travel-%tky

                        %state_area = 'VALIDATE_AGENCY'

                        %msg        = NEW zcm_rap_####(

                                            severity =
if_abap_behv_message=>severity-error

                                            textid   = zcm_rap_####=>agency_unknown

                                            agencyid = travel-AgencyID )

                        %element-AgencyID = if_abap_behv=>mk-on )

          TO reported-travel.

      ENDIF.

    ENDLOOP.
```

9. Vous allez maintenant implémenter la validation **validateCustomer** qui vérifie les éléments fournis **CustomerID** lors de *la sauvegarde* – lorsqu'ils sont modifiés dans une instance existante ou qu'une nouvelle instance est créée.

La mise en œuvre de la méthode correspondante **validateCustomer** suit une approche très similaire à celle de la méthode **validateAgency**.

Vous lirez le **CustomerID**, vérifierez son existence sur la table de base de données et générerez des messages d'état si une valeur non valide a été fournie.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **validateCustomer**.

```
" Read relevant travel instance data

    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel
```

```
        FIELDS ( CustomerID ) WITH CORRESPONDING #( keys )

      RESULT DATA(travels).


    DATA customers TYPE SORTED TABLE OF /dmo/customer WITH UNIQUE KEY customer_id.


    " Optimization of DB select: extract distinct non-initial customer IDs

    customers = CORRESPONDING #( travels DISCARDING DUPLICATES MAPPING customer_id
= CustomerID EXCEPT * ).

    DELETE customers WHERE customer_id IS INITIAL.

    IF customers IS NOT INITIAL.

      " Check if customer ID exist

      SELECT FROM /dmo/customer FIELDS customer_id

        FOR ALL ENTRIES IN @customers

        WHERE customer_id = @customers-customer_id

        INTO TABLE @DATA(customers_db).

    ENDIF.


    " Raise msg for non existing and initial customerID

    LOOP AT travels INTO DATA(travel).

      " Clear state messages that might exist

      APPEND VALUE #(  %tky        = travel-%tky

                       %state_area = 'VALIDATE_CUSTOMER' )

        TO reported-travel.


      IF travel-CustomerID IS INITIAL OR NOT line_exists( customers_db[
customer_id = travel-CustomerID ] ).

        APPEND VALUE #(  %tky = travel-%tky ) TO failed-travel.
```

```
        APPEND VALUE #(  %tky        = travel-%tky

                         %state_area = 'VALIDATE_CUSTOMER'

                         %msg        = NEW zcm_rap_####(

                                           severity   =
if_abap_behv_message=>severity-error

                                           textid     =
zcm_rap_####=>customer_unknown

                                           customerid = travel-CustomerID )

                         %element-CustomerID = if_abap_behv=>mk-on )

          TO reported-travel.

      ENDIF.

    ENDLOOP.
```

10. Vous allez maintenant mettre en œuvre la troisième et dernière validation **validateDates** qui valide la **BeginDate** et la **EndDate** du voyage.

La mise en œuvre de la méthode correspondante **validateDates** suit une approche très similaire à celle des deux autres méthodes **validateAgency** et **validateCustomer**.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **validateDates**.

```
" Read relevant travel instance data

    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel

        FIELDS ( TravelID BeginDate EndDate ) WITH CORRESPONDING #( keys )

      RESULT DATA(travels).


    LOOP AT travels INTO DATA(travel).

      " Clear state messages that might exist

      APPEND VALUE #(  %tky        = travel-%tky
```

```
                        %state_area = 'VALIDATE_DATES' )
        TO reported-travel.


      IF travel-EndDate < travel-BeginDate.
        APPEND VALUE #( %tky = travel-%tky ) TO failed-travel.
        APPEND VALUE #( %tky               = travel-%tky
                        %state_area        = 'VALIDATE_DATES'
                        %msg               = NEW zcm_rap_####(
                                                 severity  =
if_abap_behv_message=>severity-error
                                                 textid    =
zcm_rap_####=>date_interval
                                                 begindate = travel-BeginDate
                                                 enddate   = travel-EndDate
                                                 travelid  = travel-TravelID )
                        %element-BeginDate = if_abap_behv=>mk-on
                        %element-EndDate   = if_abap_behv=>mk-on ) TO reported-
travel.


      ELSEIF travel-BeginDate < cl_abap_context_info=>get_system_date( ).
        APPEND VALUE #( %tky               = travel-%tky ) TO failed-travel.
        APPEND VALUE #( %tky               = travel-%tky
                        %state_area        = 'VALIDATE_DATES'
                        %msg               = NEW zcm_rap_####(
                                                 severity  =
if_abap_behv_message=>severity-error
                                                 textid    =
zcm_rap_####=>begin_date_before_system_date
                                                 begindate = travel-BeginDate )
```

```
                        %element-BeginDate = if_abap_behv=>mk-on ) TO reported-
travel.

      ENDIF.

    ENDLOOP.
```

11. Passons maintenant à l'implémentation des déterminations. Vous allez d'abord implémenter la détermination **setInitialStatus** qui définit le *statut de voyage* en « ouvert » lors de la création d'une nouvelle instance *de voyage.*

L'implémentation lira l'état de voyage pour toutes les clés fournies et définira l'état sur ouvert pour celles qui n'ont pas encore de statut.

Remarque :
les déterminations doivent être idempotentes, ce qui signifie que le résultat peut ne pas différer même si elles sont exécutées plusieurs fois pour la même clé.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **setInitialStatus**.

```
" Read relevant travel instance data

    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel

        FIELDS ( TravelStatus ) WITH CORRESPONDING #( keys )

      RESULT DATA(travels).


    " Remove all travel instance data with defined status

    DELETE travels WHERE TravelStatus IS NOT INITIAL.

    CHECK travels IS NOT INITIAL.


    " Set default travel status

    MODIFY ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

    ENTITY Travel

      UPDATE
```

```
        FIELDS ( TravelStatus )

        WITH VALUE #( FOR travel IN travels

                      ( %tky         = travel-%tky

                        TravelStatus = travel_status-open ) )

    REPORTED DATA(update_reported).


    reported = CORRESPONDING #( DEEP update_reported ).
```

12. Vous allez maintenant implémenter la détermination **calculateTravelID** utilisée comme exemple pour illustrer une détermination lors de *l'enregistrement* et fournir un *identifiant de voyage* lisible . Notez cependant que la clé de l'objet métier de voyage reste **TravelUUID**.

Au lieu d'utiliser une plage de nombres, l'implémentation actuelle récupère simplement l' *identifiant de voyage* le plus élevé de la table de base de données et augmente sa valeur de *1.*
Veuillez noter que cette approche ne garantit pas des identifiants uniques ni sans interruption.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **calculateTravelID**.

```
" Please note that this is just an example for calculating a field during
_onSave_.

    " This approach does NOT ensure for gap free or unique travel IDs! It just
helps to provide a readable ID.

    " The key of this business object is a UUID, calculated by the framework.



    " check if TravelID is already filled

    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel

        FIELDS ( TravelID ) WITH CORRESPONDING #( keys )

      RESULT DATA(travels).
```

```
" remove lines where TravelID is already filled.
DELETE travels WHERE TravelID IS NOT INITIAL.

" anything left ?
CHECK travels IS NOT INITIAL.

" Select max travel ID
SELECT SINGLE
    FROM  zrap_atrav_####
    FIELDS MAX( travel_id ) AS travelID
    INTO @DATA(max_travelid).

" Set the travel ID
MODIFY ENTITIES OF zi_rap_travel_#### IN LOCAL MODE
ENTITY Travel
  UPDATE
    FROM VALUE #( FOR travel IN travels INDEX INTO i (
      %tky              = travel-%tky
      TravelID          = max_travelid + i
      %control-TravelID = if_abap_behv=>mk-on ) )
REPORTED DATA(update_reported).

reported = CORRESPONDING #( DEEP update_reported ).
```

13. Maintenant, vous allez implémenter la détermination **calculateTotalPrice** qui est une détermination lors de la modification des champs **BookingFee** et **CurrencyCode**– c'est-à-dire qu'elle est déclenchée à chaque fois que ces champs sont modifiés.

L'implémentation appellera l'action interne **recalcTotalPrice** pour toutes les clés de voyage fournies.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **calculateTotalPrice**. N'oubliez pas de remplacer l'espace réservé **####** par le suffixe de votre choix.

```
MODIFY ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY travel

        EXECUTE recalcTotalPrice

        FROM CORRESPONDING #( keys )

      REPORTED DATA(execute_reported).


    reported = CORRESPONDING #( DEEP execute_reported ).
```

L'action interne **recalcTotalPrice** calcule le *prix total* en parcourant toutes les réservations, en collectant tous *les prix des vols* (**FlightPrice**) et en ajoutant les *frais de réservation* (**BookingFee**). En cas de codes de devises différents, une conversion de devises est également effectuée.

La lecture des instances *de réservation* pour une instance *de voyage* s'effectue via une *lecture par association*.

L'implémentation utilise l'instruction ABAP **collect** pour créer une table interne contenant tous les codes de devise présents et additionner les valeurs des codes existants. Cette opération est effectuée pour toutes les instances *de réservation* appartenant à une instance *de voyage*. Cette table interne est ensuite ajoutée à la table **TotalPrice**, via une procédure AMDP (ABAP Managed Database Procedure) pour convertir les montants avec différents codes de devise. Enfin, le résultat **TotalPrice** est mis à jour via EML.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **recalcTotalPrice**.

```
TYPES: BEGIN OF ty_amount_per_currencycode,

             amount        TYPE /dmo/total_price,

             currency_code TYPE /dmo/currency_code,

           END OF ty_amount_per_currencycode.
```

```
   DATA: amount_per_currencycode TYPE STANDARD TABLE OF
ty_amount_per_currencycode.


   " Read all relevant travel instances.

   READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

         ENTITY Travel

            FIELDS ( BookingFee CurrencyCode )

            WITH CORRESPONDING #( keys )

         RESULT DATA(travels).


   DELETE travels WHERE CurrencyCode IS INITIAL.


   LOOP AT travels ASSIGNING FIELD-SYMBOL(<travel>).

      " Set the start for the calculation by adding the booking fee.

      amount_per_currencycode = VALUE #( ( amount        = <travel>-BookingFee

                                           currency_code = <travel>-CurrencyCode )
).

    " Read all associated bookings and add them to the total price.

    READ ENTITIES OF ZI_RAP_Travel_#### IN LOCAL MODE

       ENTITY Travel BY \_Booking

          FIELDS ( FlightPrice CurrencyCode )

        WITH VALUE #( ( %tky = <travel>-%tky ) )

        RESULT DATA(bookings).

    LOOP AT bookings INTO DATA(booking) WHERE CurrencyCode IS NOT INITIAL.

        COLLECT VALUE ty_amount_per_currencycode( amount        = booking-
FlightPrice

                                                  currency_code = booking-
CurrencyCode ) INTO amount_per_currencycode.
```

```
    ENDLOOP.


   CLEAR <travel>-TotalPrice.

     LOOP AT amount_per_currencycode INTO DATA(single_amount_per_currencycode).

        " If needed do a Currency Conversion

        IF single_amount_per_currencycode-currency_code = <travel>-CurrencyCode.

          <travel>-TotalPrice += single_amount_per_currencycode-amount.

        ELSE.

          /dmo/cl_flight_amdp=>convert_currency(

             EXPORTING

               iv_amount                   =  single_amount_per_currencycode-
amount

               iv_currency_code_source     =  single_amount_per_currencycode-
currency_code

               iv_currency_code_target     =  <travel>-CurrencyCode

               iv_exchange_rate_date       =
cl_abap_context_info=>get_system_date( )

             IMPORTING

               ev_amount                   = DATA(total_booking_price_per_curr)

            ).

          <travel>-TotalPrice += total_booking_price_per_curr.

        ENDIF.

      ENDLOOP.

    ENDLOOP.


   " write back the modified total_price of travels

    MODIFY ENTITIES OF ZI_RAP_Travel_#### IN LOCAL MODE

      ENTITY travel
```

```
        UPDATE FIELDS ( TotalPrice )

        WITH CORRESPONDING #( travels ).
```

14. Vous allez maintenant implémenter la méthode **get_features** utilisée pour le contrôle dynamique des fonctionnalités spécifiques à l'instance. Elle attend dans la table de résultats une entrée pour chaque clé demandée, spécifiant si la fonctionnalité associée est activée ou désactivée selon l'état de l'instance.

Dans le cas présent, les actions **acceptTravel** et **rejectTravel** sont basculées en fonction de l'état actuel.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **get_features**.

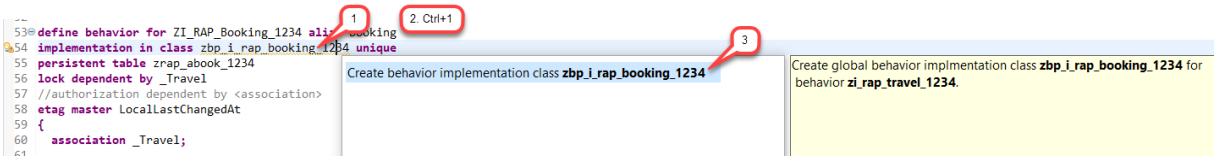
```
" Read the travel status of the existing travels

    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel

        FIELDS ( TravelStatus ) WITH CORRESPONDING #( keys )

      RESULT DATA(travels)

      FAILED failed.


    result =

      VALUE #(

        FOR travel IN travels

          LET is_accepted =   COND #( WHEN travel-TravelStatus = travel_status-
accepted

                                      THEN if_abap_behv=>fc-o-disabled

                                      ELSE if_abap_behv=>fc-o-enabled   )

              is_rejected =   COND #( WHEN travel-TravelStatus = travel_status-
canceled

                                      THEN if_abap_behv=>fc-o-disabled

                                      ELSE if_abap_behv=>fc-o-enabled )

          IN
```

```
            ( %tky                   = travel-%tky
              %action-acceptTravel = is_accepted
              %action-rejectTravel = is_rejected
            ) ).
```

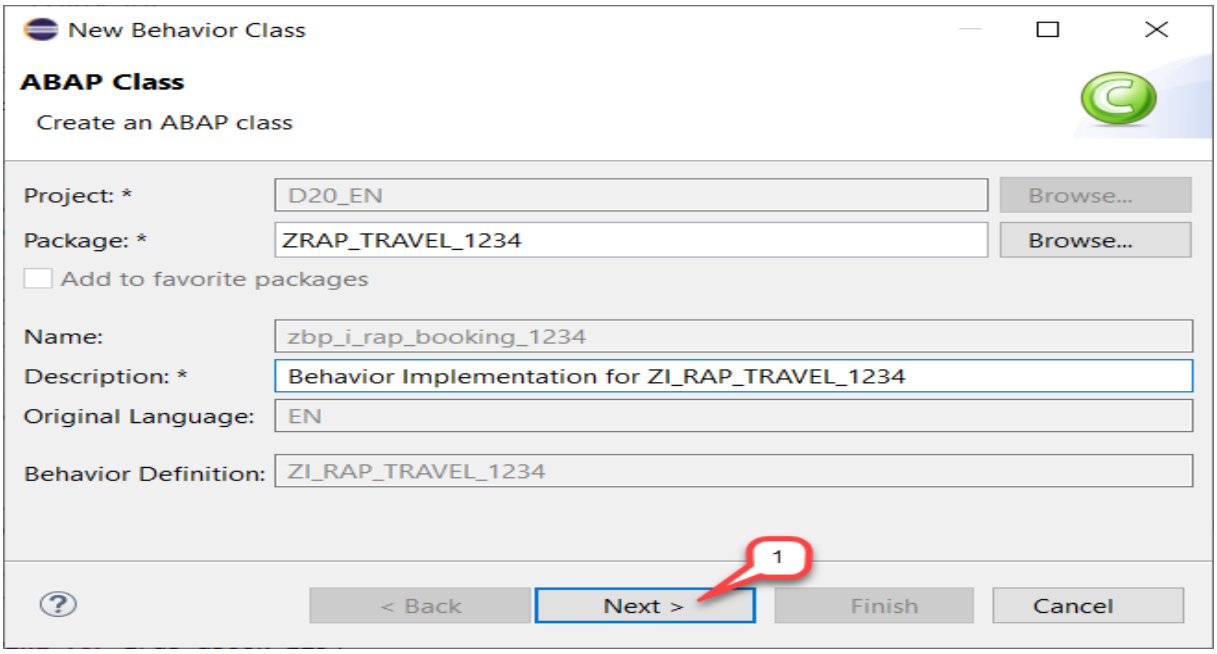
Enregistrez et activez l'implémentation du comportement.

Étape 4 : Implémenter le pool de comportements pour l'entité *Réservation*

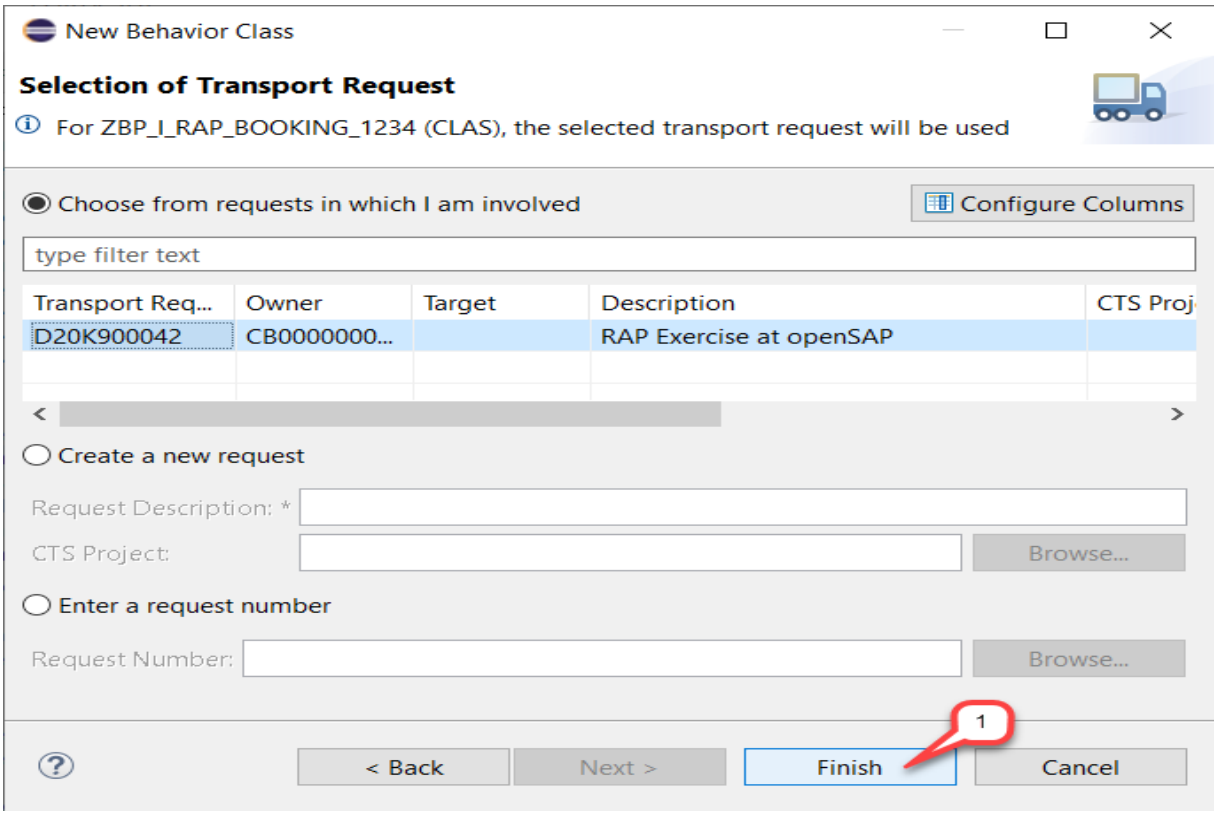
1. Revenez à la *définition du comportement* **ZI_RAP_Travel_####** de base de votre objet métier dans l'Explorateur *de projets* .
   Placez le curseur sur le nom de la classe d'implémentation du comportement **zcl_i_rap_booking_####**,
   ouvrez la boîte de dialogue *Correction rapide* avec le raccourci clavier **Ctrl+1** et choisissez **create…** pour générer l'implémentation du comportement pour l'entité Voyage.



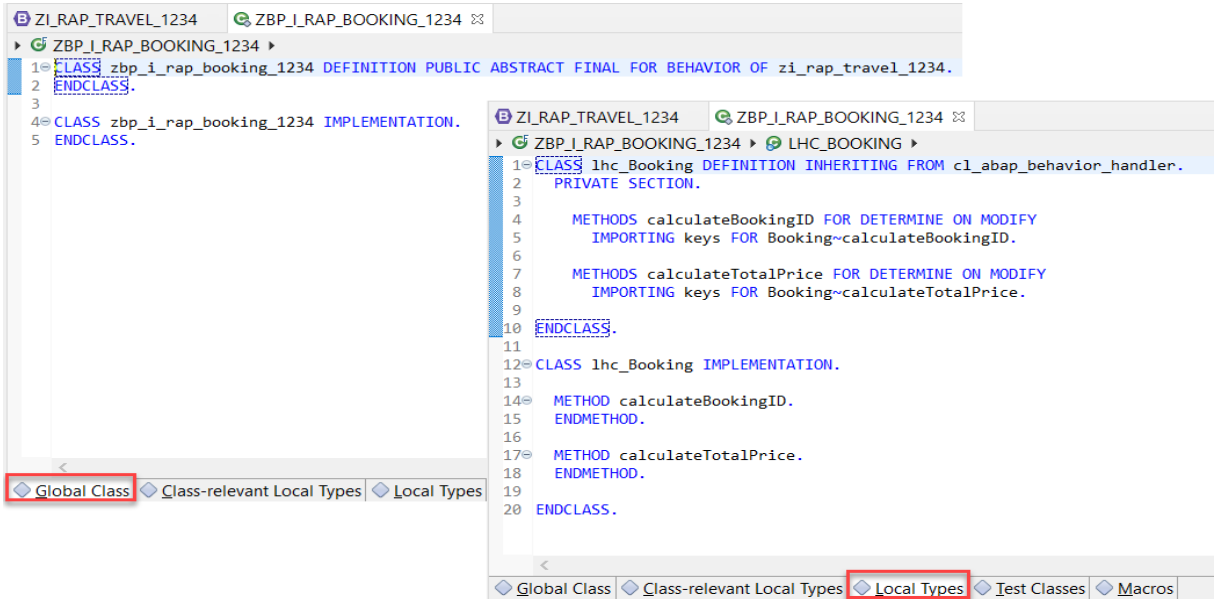
2. Dans l'assistant qui apparaît, conservez les champs par défaut et choisissez **Suivant >**.



3. Attribuez une demande de transport et choisissez **Terminer**.



Le squelette de l'implémentation du comportement est généré et affiché dans l'éditeur.



4. Commencez par implémenter la détermination **calculateBookingID** effectuée sur *Modify* lors de la création d'une nouvelle instance. Pour cela, insérez l'extrait de code ci-dessous dans la méthode **calculateBookingID**.

```
DATA max_bookingid TYPE /dmo/booking_id.

    DATA update TYPE TABLE FOR UPDATE ZI_RAP_Travel_####\\Booking.



    " Read all travels for the requested bookings.

    " If multiple bookings of the same travel are requested, the travel is
returned only once.

    READ ENTITIES OF ZI_RAP_Travel_#### IN LOCAL MODE

    ENTITY Booking BY \_Travel

      FIELDS ( TravelUUID )

      WITH CORRESPONDING #( keys )

      RESULT DATA(travels).



    " Process all affected Travels. Read respective bookings, determine the max-id
and update the bookings without ID.

    LOOP AT travels INTO DATA(travel).

      READ ENTITIES OF ZI_RAP_Travel_#### IN LOCAL MODE

        ENTITY Travel BY \_Booking

          FIELDS ( BookingID )

        WITH VALUE #( ( %tky = travel-%tky ) )

        RESULT DATA(bookings).



      " Find max used BookingID in all bookings of this travel

      max_bookingid ='0000'.

      LOOP AT bookings INTO DATA(booking).
```

```
      IF booking-BookingID > max_bookingid.
        max_bookingid = booking-BookingID.
      ENDIF.
    ENDLOOP.

    " Provide a booking ID for all bookings that have none.
    LOOP AT bookings INTO booking WHERE BookingID IS INITIAL.
      max_bookingid += 10.
      APPEND VALUE #( %tky      = booking-%tky
                      BookingID = max_bookingid
                    ) TO update.
    ENDLOOP.
  ENDLOOP.

  " Update the Booking ID of all relevant bookings
  MODIFY ENTITIES OF ZI_RAP_Travel_#### IN LOCAL MODE
  ENTITY Booking
    UPDATE FIELDS ( BookingID ) WITH update
  REPORTED DATA(update_reported).

  reported = CORRESPONDING #( DEEP update_reported ).
```

On utilise d'abord une *méthode de lecture par association* pour lire l'en-tête *de voyage* associé . Pour chaque instance *de voyage*, toutes les réservations sont lues afin de trouver la plus élevée **BookingID** et de calculer la valeur **BookingID** pour les entrées qui n'en contiennent pas encore.

5. La détermination **calculateTotalPrice** appelle l'action interne **recalcTotalPrice** que nous avons définie et mise en œuvre au niveau racine.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **calculateTotalPrice**.

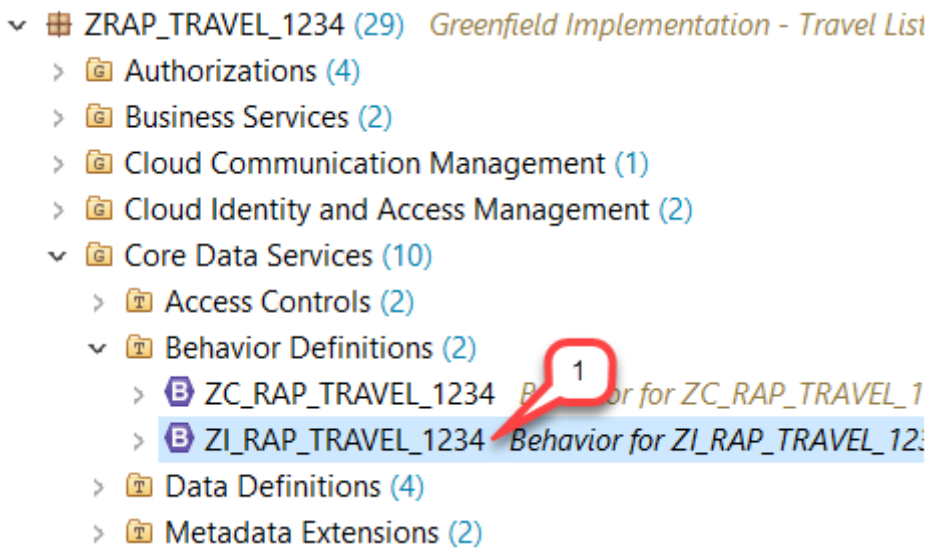
```
" Read all travels for the requested bookings.

    " If multiple bookings of the same travel are requested, the travel is
returned only once.

    READ ENTITIES OF ZI_RAP_Travel_#### IN LOCAL MODE

    ENTITY Booking BY \_Travel

      FIELDS ( TravelUUID )

      WITH CORRESPONDING #( keys )

      RESULT DATA(travels)

      FAILED DATA(read_failed).


    " Trigger calculation of the total price

    MODIFY ENTITIES OF ZI_RAP_Travel_#### IN LOCAL MODE

    ENTITY Travel

      EXECUTE recalcTotalPrice

      FROM CORRESPONDING #( travels )

    REPORTED DATA(execute_reported).


    reported = CORRESPONDING #( DEEP execute_reported ).
```

Enregistrez et activez l'implémentation du comportement.

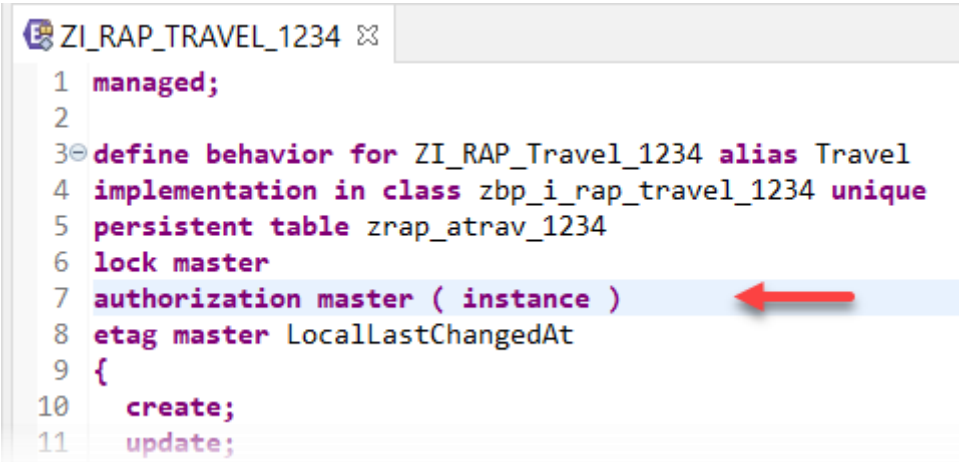
Étape 5 : Définir le maître d'autorisation

Vous allez maintenant ajouter un contrôle d'autorisation à votre objet métier.
Pour cela, vous devrez d'abord améliorer la *définition de comportement* de base en conséquence.

1. Dans l'Explorateur *de projets*, ouvrez la *définition de comportement* **ZI_RAP_Travel_####** de base où **####**se trouve le suffixe que vous avez choisi.



2. Supprimez le commentaire **authorization master (instance)** de la ligne d'instruction pour déclarer le nœud racine *de voyage* comme maître d'autorisation. Les demandes d'autorisation spécifiques à une instance sont dirigées vers l'implémentation du maître d'autorisation.



3. Maintenant, faites défiler jusqu'à l' entité *de réservation* et supprimez le commentaire de la ligne dépendante de l'autorisation et spécifiez l'association **_Travel**.

```
authorization dependent by _Travel
```

Cela déclare l'entité *de réservation* comme dépendante de l'autorisation. Les demandes de modification sur l'entité *de réservation* sont dirigées vers le maître d'autorisation suivant l'association fournie.

Votre code source devrait ressembler à ceci :

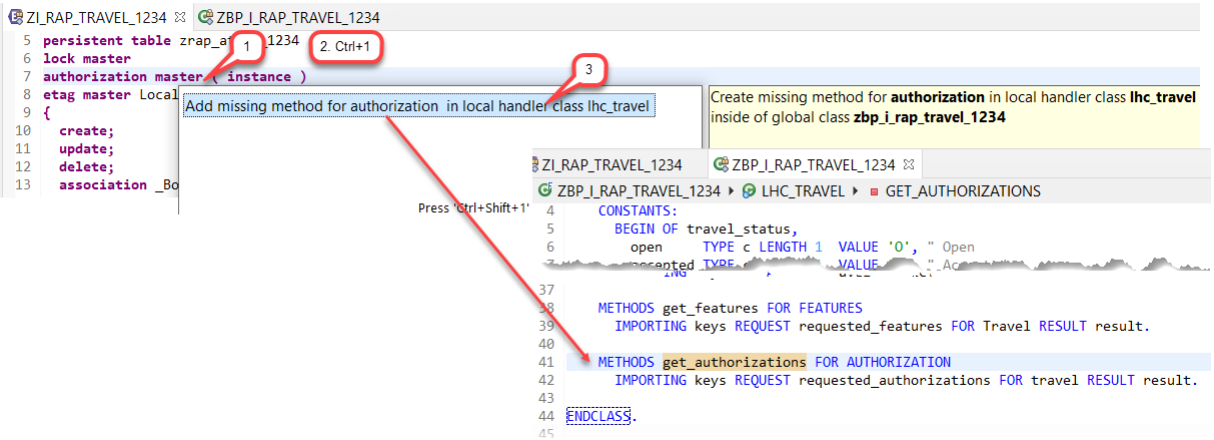
```
53 define behavior for ZI_RAP_Booking_1234 alias Booking
54 implementation in class zbp_i_rap_booking_1234 unique
55 persistent table zrap_abook_1234
56 lock dependent by _Travel
57 authorization dependent by _Travel
58 etag master LocalLastChangedAt
59 {
60   association _Travel;
61
62   update;
```

4. Enregistrez et activez la définition du comportement.

Étape 6 : Implémenter le maître d'autorisation

Vous allez maintenant implémenter le maître d'autorisation dans l'implémentation du comportement de l'entité *de voyage* **ZI_RAP_Travel_####**.

1. Tout d'abord, restez dans votre définition de comportement BO de base **ZI_RAP_Travel_####**.
   Placez le curseur sur le mot-clé **master** de la déclaration principale d'autorisation et utilisez la *correction rapide* (**Ctrl+1**) pour ajouter la nouvelle déclaration de méthode à l'implémentation du comportement.



2. Comme vous ne pouvez pas influencer les autorisations réelles dans l'environnement ABAP d'essai, vous devez simuler la présence ou l'absence d'une autorisation spécifique, par exemple l'autorisation de mise à jour.
   Cela se fait via des méthodes d'assistance que nous devons d'abord déclarer.

Pour cela, insérez l'extrait de code fourni ci-dessous à la fin de la définition de la classe de gestionnaire local **lhc_Travel** de l'implémentation du comportement de l'entité *de voyage*.

```
METHODS is_update_granted IMPORTING has_before_image      TYPE abap_bool

                                    overall_status        TYPE
/dmo/overall_status

                          RETURNING VALUE(update_granted) TYPE abap_bool.


   METHODS is_delete_granted IMPORTING has_before_image      TYPE abap_bool

                                    overall_status        TYPE
/dmo/overall_status

                          RETURNING VALUE(delete_granted) TYPE abap_bool.


   METHODS is_create_granted RETURNING VALUE(create_granted) TYPE abap_bool.
```

Votre code source devrait ressembler à ceci :

```
ZI_RAP_TRAVEL_1234   ZBP_I_RAP_TRAVEL_1234
ZBP_I_RAP_TRAVEL_1234 ▸ LHC_TRAVEL ▸
1  CLASS lhc_Travel DEFINITION INHERITING FROM cl_abap_behavior_handler.
2    PRIVATE SECTION.
3
4      CONSTANTS:
5        BEGIN OF travel_status,
6          open     TYPE c LENGTH 1  VALUE 'O', " Open
7          accepted TYPE c LENGTH 1  VALUE 'A', " Accepted
35     METHODS  recalcTotalPrice  FOR  ...
36       IMPORTING keys FOR ACTION Travel~recalcTotalPrice.
37
38     METHODS get_features FOR FEATURES
39       IMPORTING keys REQUEST requested_features FOR Travel RESULT result.
40
41     METHODS get_authorizations FOR AUTHORIZATION
42       IMPORTING keys REQUEST requested_authorizations FOR travel RESULT result.
43
44     METHODS is_update_granted IMPORTING has_before_image      TYPE abap_bool
45                                         overall_status         TYPE /dmo/overall_status
46                               RETURNING VALUE(update_granted) TYPE abap_bool.
47
48     METHODS is_delete_granted IMPORTING has_before_image      TYPE abap_bool
49                                         overall_status         TYPE /dmo/overall_status
50                               RETURNING VALUE(delete_granted) TYPE abap_bool.
51
52     METHODS is_create_granted RETURNING VALUE(create_granted) TYPE abap_bool.
53
54 ENDCLASS.
55
56 CLASS lhc_Travel IMPLEMENTATION.
57
```

3. Utilisez le correctif rapide ADT (**Ctrl+1**) pour générer les implémentations des quatre méthodes.

```
38     METHODS get_features FOR FEATURES
39       IMPORTING keys REQUEST r  1  ste  2. Ctrl+1  FOR Travel RESULT result.
40
41     METHODS get_authorizations FOR AUTHORIZATION
42       IMPORTING keys REQUEST r
43
44     METHODS is_update_granted
45
46
47
48     METHODS is_delete_granted
49
50
51
52     METHODS is_create_granted
53
54 ENDCLASS.
55
56 CLASS lhc_Travel IMPLEMENTATIO
<
Global Class | Class-relevant Local Types

Add ABAP Doc
Rename get_authorizations (Ctrl+2
Add 4 unimplemented methods   3
Delete get_authorizations

Creates empty method bodies for all unimplemented methods of class lhc_travel.

lhc_travel
  - get_authorizations
  - is_create_granted
  - is_delete_granted
  - is_update_granted

Press 'Tab' from proposal table or click f

383  METHOD get_authorizations.
384
385  ENDMETHOD.
386
387  METHOD is_create_granted.
388
389  ENDMETHOD.
390
391  METHOD is_delete_granted.
392
393  ENDMETHOD.
394
395  METHOD is_update_granted.
396
397  ENDMETHOD.
398
399 ENDCLASS.
```

Vous allez maintenant implémenter les différentes méthodes. Les classes d'assistance effectueront les contrôles d'autorité à l'aide de l'objet d'autorité ZOSTAT**####**défini la semaine précédente.

4. Tout d'abord, implémentez la méthode d'assistance **is_create_granted** qui vérifie si l'opération *de création* est autorisée. Cette opération correspond à l'activité **01**.

À des fins de test, nous avons défini le résultat en **true** dans tous les cas ». Ce paramètre peut bien sûr être ajusté pour tester différentes situations.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **is_create_granted**.

```
AUTHORITY-CHECK OBJECT 'ZOSTAT####'

      ID 'ZOSTAT####' DUMMY

      ID 'ACTVT' FIELD '01'.

    create_granted = COND #( WHEN sy-subrc = 0 THEN abap_true ELSE abap_false ).

    " Simulate full access - for testing purposes only! Needs to be removed for a
productive implementation.

    create_granted = abap_true.
```

5. Implémentez maintenant la méthode d'assistance **is_update_granted**qui vérifie si l'opération *de mise à jour* est autorisée. Cette opération correspond à l'activité **02**. Pour cela, insérez l'extrait de code ci-dessous dans la méthode **is_update_granted**.

```
IF has_before_image = abap_true.

       AUTHORITY-CHECK OBJECT 'ZOSTAT####'

         ID 'ZOSTAT####' FIELD overall_status

         ID 'ACTVT' FIELD '02'.

     ELSE.

       AUTHORITY-CHECK OBJECT 'ZOSTAT####'

         ID 'ZOSTAT####' DUMMY

         ID 'ACTVT' FIELD '02'.

     ENDIF.
```

```
     update_granted = COND #( WHEN sy-subrc = 0 THEN abap_true ELSE abap_false ).



     " Simulate full access - for testing purposes only! Needs to be removed for a
productive implementation.

     update_granted = abap_true.
```

6. Implémentez maintenant la méthode d'assistance **is_delete_granted** qui vérifie si l'opération *de suppression* est autorisée. Cette opération correspond à l'activité **06**.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **is_delete_granted**.

```
IF has_before_image = abap_true.

      AUTHORITY-CHECK OBJECT 'ZOSTAT####'

        ID 'ZOSTAT####' FIELD overall_status

        ID 'ACTVT' FIELD '06'.

    ELSE.

      AUTHORITY-CHECK OBJECT 'ZOSTAT####'

        ID 'ZOSTAT####' DUMMY

        ID 'ACTVT' FIELD '06'.

    ENDIF.

    delete_granted = COND #( WHEN sy-subrc = 0 THEN abap_true ELSE abap_false ).



    " Simulate full access - for testing purposes only! Needs to be removed for a
productive implementation.

    delete_granted = abap_true.
```

7. Vous pouvez maintenant implémenter la **get_authorizations**méthode.
Elle simule le cas d'utilisation d'un champ modifiable (**TravelStatus**ici) pour contrôler l'autorisation. Par conséquent, l'implémentation lit l'image précédente de la persistance active et utilise cette valeur lors de l'appel des méthodes d'assistance.

Pour cela, insérez l'extrait de code ci-dessous dans la méthode **get_authorizations**.

```
DATA: has_before_image    TYPE abap_bool,
```

```
          is_update_requested TYPE abap_bool,

          is_delete_requested TYPE abap_bool,

          update_granted      TYPE abap_bool,

          delete_granted      TYPE abap_bool.


  DATA: failed_travel LIKE LINE OF failed-travel.


  " Read the existing travels

    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel

        FIELDS ( TravelStatus ) WITH CORRESPONDING #( keys )

      RESULT DATA(travels)

      FAILED failed.


    CHECK travels IS NOT INITIAL.


 *   In this example the authorization is defined based on the Activity + Travel
Status

 *   For the Travel Status we need the before-image from the database. We perform
this for active (is_draft=00) as well as for drafts (is_draft=01) as we can't
distinguish between edit or new drafts

    SELECT FROM zrap_atrav_####

      FIELDS travel_uuid,overall_status

      FOR ALL ENTRIES IN @travels

      WHERE travel_uuid EQ @travels-TravelUUID

      ORDER BY PRIMARY KEY

      INTO TABLE @DATA(travels_before_image).
```

```
     is_update_requested = COND #( WHEN requested_authorizations-%update
= if_abap_behv=>mk-on OR

                                          requested_authorizations-%action-
acceptTravel = if_abap_behv=>mk-on OR

                                          requested_authorizations-%action-
rejectTravel = if_abap_behv=>mk-on OR

 *                                         requested_authorizations-%action-Prepare
= if_abap_behv=>mk-on OR

 *                                         requested_authorizations-%action-Edit
= if_abap_behv=>mk-on OR

                                          requested_authorizations-%assoc-_Booking
= if_abap_behv=>mk-on

                                     THEN abap_true ELSE abap_false ).


   is_delete_requested = COND #( WHEN requested_authorizations-%delete =
if_abap_behv=>mk-on

                                     THEN abap_true ELSE abap_false ).


   LOOP AT travels INTO DATA(travel).

       update_granted = delete_granted = abap_false.


   READ TABLE travels_before_image INTO DATA(travel_before_image)

        WITH KEY travel_uuid = travel-TravelUUID BINARY SEARCH.

       has_before_image = COND #( WHEN sy-subrc = 0 THEN abap_true ELSE abap_false
).


   IF is_update_requested = abap_true.

         " Edit of an existing record -> check update authorization

         IF has_before_image = abap_true.

           update_granted = is_update_granted( has_before_image = has_before_image
overall_status = travel_before_image-overall_status ).
```

```
          IF update_granted = abap_false.

            APPEND VALUE #( %tky        = travel-%tky

                            %msg        = NEW zcm_rap_####( severity =
if_abap_behv_message=>severity-error

                                                                textid   =
zcm_rap_####=>unauthorized )

                          ) TO reported-travel.

          ENDIF.

          " Creation of a new record -> check create authorization

        ELSE.

          update_granted = is_create_granted( ).

          IF update_granted = abap_false.

            APPEND VALUE #( %tky        = travel-%tky

                            %msg        = NEW zcm_rap_####( severity =
if_abap_behv_message=>severity-error

                                                                textid   =
zcm_rap_####=>unauthorized )

                          ) TO reported-travel.

          ENDIF.

        ENDIF.

      ENDIF.


      IF is_delete_requested = abap_true.

        delete_granted = is_delete_granted( has_before_image = has_before_image
overall_status = travel_before_image-overall_status ).

        IF delete_granted = abap_false.

          APPEND VALUE #( %tky        = travel-%tky

                          %msg        = NEW zcm_rap_####( severity =
if_abap_behv_message=>severity-error
```

```
                                                                    textid    =
zcm_rap_####=>unauthorized )

                          ) TO reported-travel.

         ENDIF.

       ENDIF.



       APPEND VALUE #( %tky = travel-%tky



                       %update              = COND #( WHEN update_granted =
abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )

                       %action-acceptTravel = COND #( WHEN update_granted =
abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )

                       %action-rejectTravel = COND #( WHEN update_granted =
abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )

 *                      %action-Prepare      = COND #( WHEN update_granted =
abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )

 *                      %action-Edit         = COND #( WHEN update_granted =
abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )

                       %assoc-_Booking      = COND #( WHEN update_granted =
abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )



                       %delete              = COND #( WHEN delete_granted =
abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )

                     )

         TO result.

     ENDLOOP.
```

Les différentes options de la **requested_authorizations**structure sont associées aux valeurs d'activité (*créer* , *mettre à jour* et *supprimer*) définies dans l'objet d'autorisation. Celles-ci sont ensuite vérifiées pour chaque instance *de déplacement* .

Le tableau des résultats est établi en fonction du résultat des contrôles d'autorisation respectifs.

8. Enregistrez et activez l'implémentation du comportement.

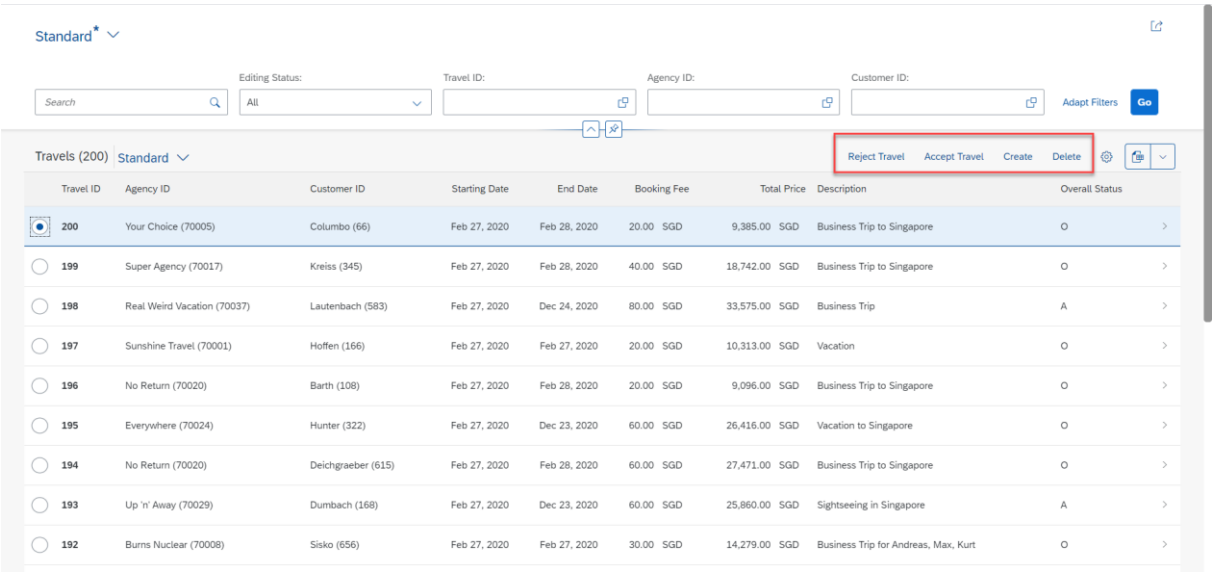
Étape 7 : Prévisualisez l'application Voyage améliorée et testez-la

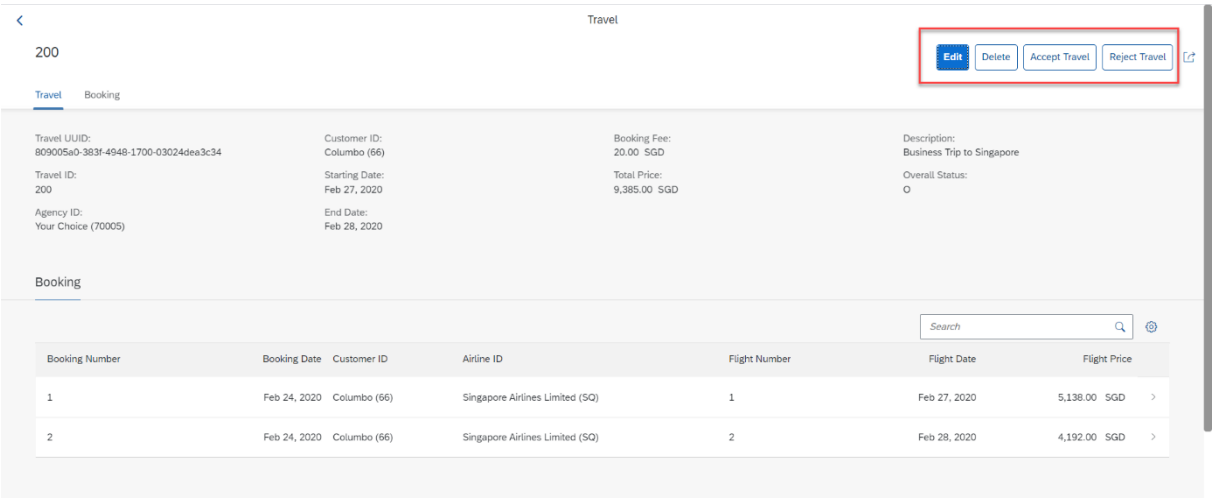
Vous avez maintenant terminé les implémentations de comportement et pouvez exécuter et vérifier les éléments améliorés de l'application de voyage SAP Fiori.

1. Démarrez l'application *Voyage* dans votre liaison de service **ZUI_RAP_TRAVEL_O2_####**(où **####**se trouve le suffixe choisi) ou actualisez- **F5**la ( ) dans le navigateur.
   Fournissez vos identifiants d'utilisateur ABAP si nécessaire.

Vous verrez d'abord les deux nouvelles actions « *Accepter le voyage* » et « *Rejeter le voyage* » à côté des opérations de création, de mise à jour et de suppression. Ces deux actions sont activées en fonction du *statut* de l'instance de voyage sélectionnée.

Appuyez sur **Go** sur l'interface utilisateur pour charger les données du back-end.





2. N'hésitez pas à expérimenter avec votre application.
   Par exemple :

- Créez un nouvel enregistrement de voyage.
  Fournissez des valeurs incohérentes pour l' *ID d'agence* , l' *ID client* et la *date de début* et/ou *de fin* .
  Lorsque vous cliquez sur « *Enregistrer* » , les messages d'erreur correspondants s'affichent.

- Corrigez les valeurs et cliquez sur « *Enregistrer* » .
  Les validations sont désormais réussies. Vous constatez également que les déterminations ont été effectuées. L' *identifiant de voyage* a été calculé et le *statut de réservation* est revenu à la valeur par défaut.

- Utilisez les actions *Accepter* et *Rejeter* pour vérifier le contrôle des fonctionnalités spécifiques à l'instance.

- Sur la page de l'objet Voyage :
  dans la table « *Réservation* » , cliquez sur « *Créer* » pour créer une instance *de réservation* pour ce voyage. Saisissez des entrées valides et cliquez sur « *Enregistrer* » .
  Un retour en arrière indique que l'instance d'en-tête n'a pas été rechargée et que l'ancien *prix total* est donc affiché. Cela nécessiterait la définition d'un effet secondaire que nous n'aborderons pas dans cette session. Une actualisation manuelle de l'interface utilisateur affiche le prix total mis à jour.

- Vous pouvez cliquer sur *supprimer* pour supprimer l'instance de voyage.

Résumé

Dans cette unité, vous avez appris :

- Comment créer la classe de pool de comportements BO et comment implémenter la logique spécifique à l'objet métier pour votre application dans ABAP à l'aide de la détermination, des validations et des actions.
- Comment le contrôle dynamique des fonctionnalités améliore l'expérience utilisateur et comment le mettre en œuvre.
- Comment mettre en œuvre le maître d'autorisation.

4.6 Activation de la gestion du projet (Unité 6)

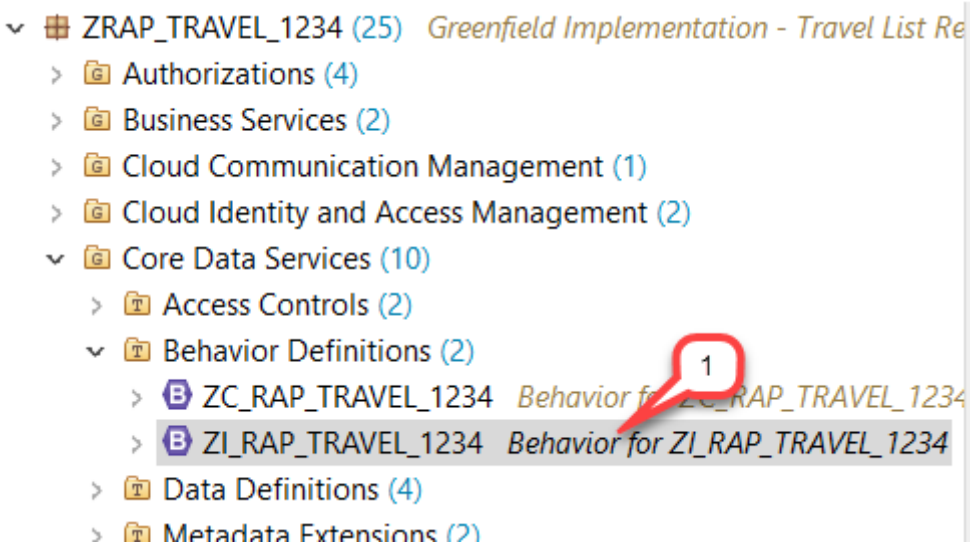
Objectif :

Vous allez maintenant activer la gestion des brouillons dans votre objet métier et tester le résultat dans votre application de voyage basée sur les éléments SAP Fiori.

Étape 1 : Activer la gestion des brouillons dans l'objet métier de base

Vous allez maintenant activer la gestion des brouillons pour l'objet métier géré (BO) avec quelques ajouts dans la définition du comportement.

1. Dans l'Explorateur *de projets* , accédez à votre package et ouvrez la définition du comportement de base **ZI_RAP_Travel_####**



2. Ajoutez l'ajout **with draft;**après le **managed;**mot-clé dans la section d'en-tête pour activer la gestion des brouillons pour votre objet métier.

```
with draft;
```

```
*ZI_RAP_TRAVEL_1234
1 managed;
2 with draft;
3
4 define behavior for ZI_RAP_Travel_1234 alias Travel
5 implementation in class zbp_i_rap_travel_1234 unique
6 persistent table zrap_atrav_1234
7 lock master
```

3. Spécifiez le brouillon de table pour l'entité *de voyage* , où les brouillons de données *de voyage* seront conservés comme indiqué sur la capture d'écran.

Les tables de brouillon sont entièrement gérées par le framework RAP lors de l'exécution

Pour cela, insérez l'extrait de code ci-dessous, comme indiqué dans la capture d'écran.

```
draft table zrap_dtrav_####
```

```
*ZI_RAP_TRAVEL_1234
1 managed;
2 with draft;
3
4 define behavior for ZI_RAP_Travel_1234 alias Travel
5 implementation in class zbp_i_rap_travel_1234 unique
6 persistent table zrap_atrav_1234
7 draft table zrap_dtrav_1234
8 lock master
```

4. Faites de même pour l'entité *de réservation* , c'est-à-dire définissez la table de brouillon dans laquelle les données *de réservation* de brouillon seront conservées.

Pour cela, insérez l'extrait de code ci-dessous dans la définition du comportement *de réservation*, comme indiqué dans la capture d'écran.

```
draft table zrap_dbook_####
```

```
55 define behavior for ZI_RAP_Booking_1234 alias Booking
56 implementation in class zbp_i_rap_booking_1234 unique
57 persistent table zrap_abook_1234
58 draft table zrap_dbook_1234
59 lock dependent by _Travel
60 authorization dependent by _Travel
61 etag master LocalLastChangedAt
62 {
```

5. Vous allez maintenant créer la table de brouillons **zrap_dtrav_####**, pour stocker les données de brouillon de l' entité *de voyage*.

   La fonctionnalité ADT *Quick Fix* peut être utilisée pour générer la table de brouillons.

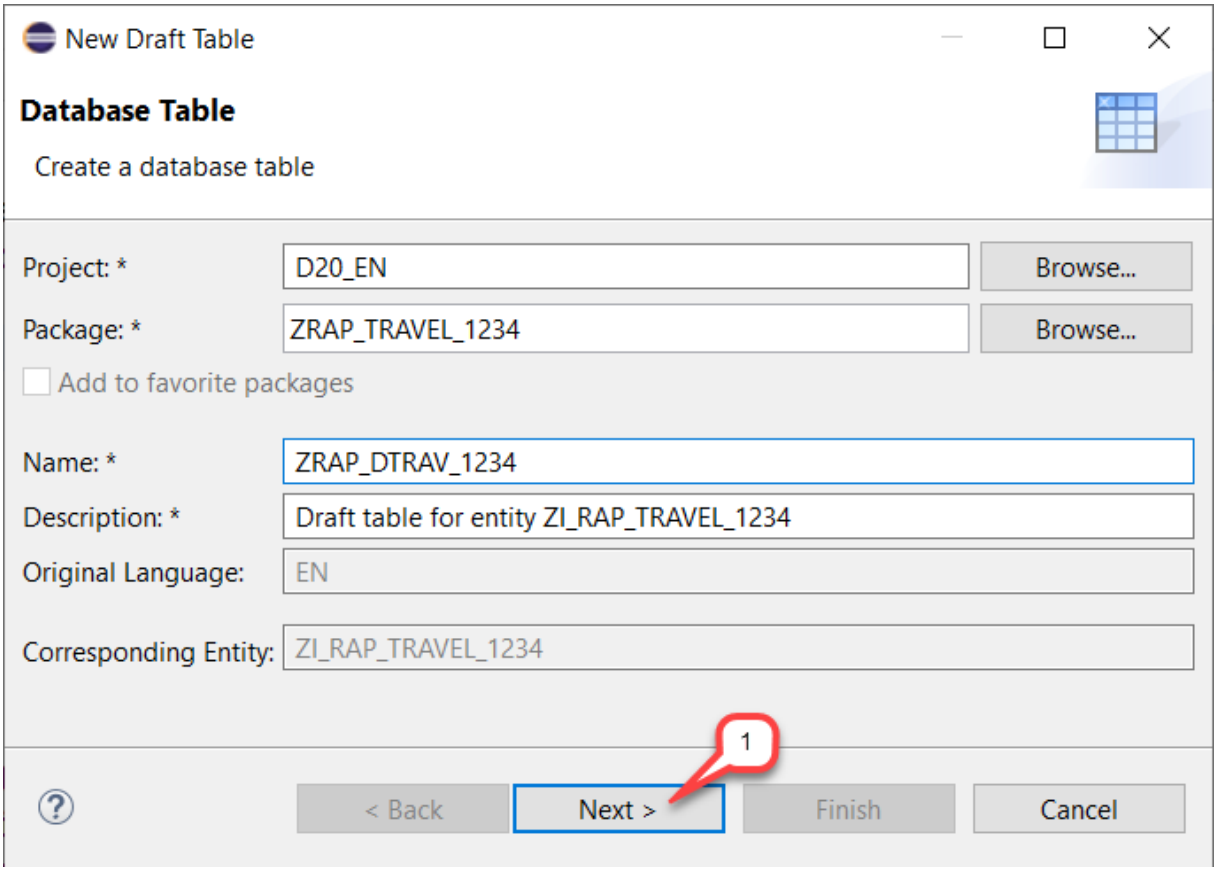
Pour cela, placez le curseur sur le nom de la table et appuyez sur **Ctrl+1** pour démarrer la boîte de dialogue *Correction rapide*.

```
*ZI_RAP_TRAVEL_1234
1 managed;
2 with draft;
3
4 define behavior for ZI_RAP_Travel_1234 alias Travel
5 implementation in class zbp_i_rap_travel_1234 unique
6 persistent table zrap_atrav_1234
7 draft table zrap_dtrav_1234
8 lock master
9 authorization master ( inst
10 etag master LocalLastChange
11 {
```

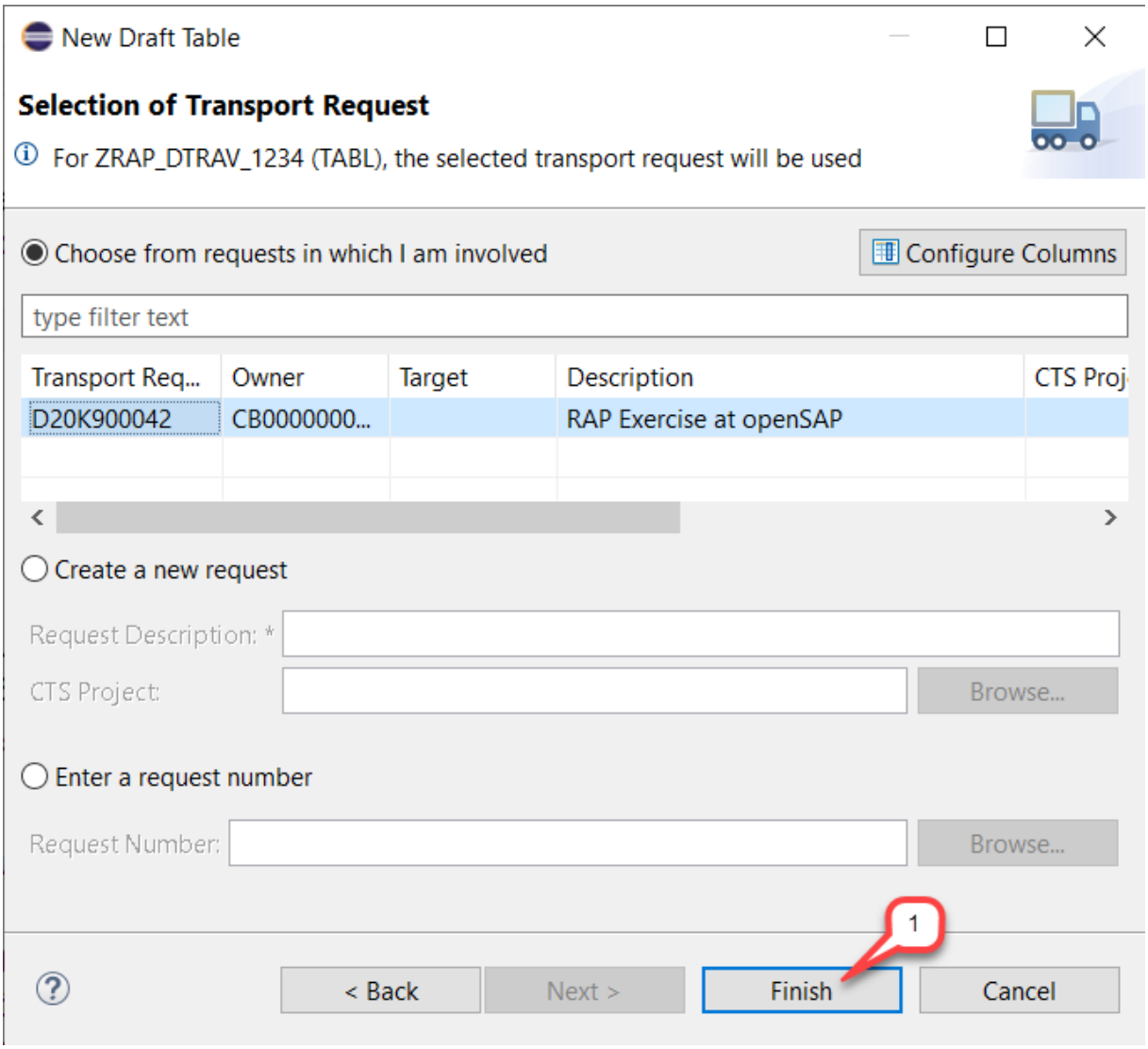
1 — 2. press Ctrl+1 — 3

Create draft table **zrap_dtrav_1234** for entity **zi_rap_travel_1234**

6. Laissez les valeurs par défaut telles qu'elles sont dans la boîte de dialogue qui apparaît et choisissez **Suivant**.


New Draft Table
Database Table
Create a database table
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Browse...
Add to favorite packages
Name: *
ZRAP_DTRAV_1234
Description: *
Draft table for entity ZI_RAP_TRAVEL_1234
Original Language:
EN
Corresponding Entity:
ZI_RAP_TRAVEL_1234
1
< Back
Next >
Finish
Cancel

7. Affectez une demande de transport et choisissez **Terminer** pour générer la table.



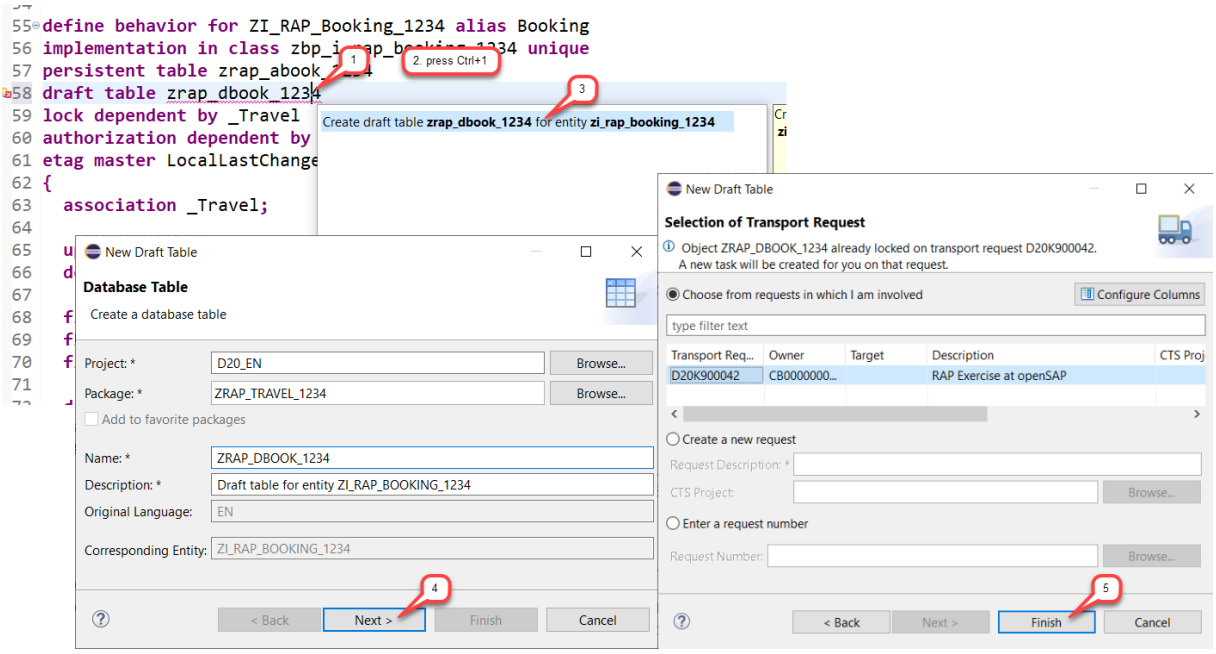
Le tableau de brouillon est généré sur la base du modèle défini et affiché dans l'éditeur approprié.

*ZI_RAP_TRAVEL_1234 | ZRAP_DTRAV_1234

```
1  @EndUserText.label : 'Draft table for entity ZI_RAP_TRAVEL_1234'
2  @AbapCatalog.enhancementCategory : #EXTENSIBLE_ANY
3  @AbapCatalog.tableCategory : #TRANSPARENT
4  @AbapCatalog.deliveryClass : #A
5  @AbapCatalog.dataMaintenance : #RESTRICTED
6  define table zrap_dtrav_1234 {
7    key mandt          : mandt not null;
8    key traveluuid     : sysuuid_x16 not null;
9    travelid           : /dmo/travel_id;
10   agencyid           : /dmo/agency_id;
11   customerid         : /dmo/customer_id;
12   begindate          : /dmo/begin_date;
13   enddate            : /dmo/end_date;
14   @Semantics.amount.currencyCode : 'zrap_dtrav_1234.currencycode'
15   bookingfee         : /dmo/booking_fee;
16   @Semantics.amount.currencyCode : 'zrap_dtrav_1234.currencycode'
17   totalprice         : /dmo/total_price;
18   currencycode       : /dmo/currency_code;
19   description        : /dmo/description;
20   travelstatus       : /dmo/overall_status;
21   createdby          : syuname;
22   createdat          : timestampl;
23   lastchangedby      : syuname;
24   lastchangedat      : timestampl;
25   locallastchangedat : timestampl;
26   "%admin"           : include sych_bdl_draft_admin_inc;
27
28 }
```

Remarque :
chaque fois que vous modifiez le modèle de données BO, vous pouvez à nouveau utiliser la *fonction de correction rapide* ADT (*Ctrl+1*) pour générer à nouveau le brouillon de définition de table. Cette opération mettra à jour la définition de table.

8. Enregistrez , activez et fermez la table.
9. Créez maintenant également le brouillon de table pour l'entité *de réservation à l'aide du correctif rapide* **zrap_dbook_####** ADT (**Ctrl+1**).


define behavior for ZI_RAP_Booking_1234 alias Booking
implementation in class zbp_i_rap_booking_1234 unique
persistent table zrap_abook_1234
draft table zrap_dbook_1234
lock dependent by _Travel
authorization dependent by
etag master LocalLastChange
{
association _Travel;
1
2. press Ctrl+1
3
Create draft table zrap_dbook_1234 for entity zi_rap_booking_1234
New Draft Table
Database Table
Create a database table
Project: *
D20_EN
Browse...
Package: *
ZRAP_TRAVEL_1234
Add to favorite packages
Name: *
ZRAP_DBOOK_1234
Description: *
Draft table for entity ZI_RAP_BOOKING_1234
Original Language:
EN
Corresponding Entity:
ZI_RAP_BOOKING_1234
4
< Back
Next >
Finish
Cancel
New Draft Table
Selection of Transport Request
Object ZRAP_DBOOK_1234 already locked on transport request D20K900042.
A new task will be created for you on that request.
Choose from requests in which I am involved
Configure Columns
type filter text
Transport Req...
Owner
Target
Description
CTS Proj
D20K900042
CB0000000...
RAP Exercise at openSAP
Create a new request
Request Description: *
CTS Project:
Enter a request number
Request Number:
5
< Back
Next >
Finish
Cancel

Le tableau de brouillon est généré sur la base du modèle défini et affiché dans l'éditeur approprié.

*ZI_RAP_TRAVEL_1234 | ZRAP_DTRAV_1234 | ZRAP_DBOOK_1234

```
1  @EndUserText.label : 'Draft table for entity ZI_RAP_BOOKING_1234'
2  @AbapCatalog.enhancementCategory : #EXTENSIBLE_ANY
3  @AbapCatalog.tableCategory : #TRANSPARENT
4  @AbapCatalog.deliveryClass : #A
5  @AbapCatalog.dataMaintenance : #RESTRICTED
6  define table zrap_dbook_1234 {
7    key mandt          : mandt not null;
8    key bookinguuid    : sysuuid_x16 not null;
9    traveluuid         : sysuuid_x16;
10   bookingid          : /dmo/booking_id;
11   bookingdate        : /dmo/booking_date;
12   customerid         : /dmo/customer_id;
13   carrierid          : /dmo/carrier_id;
14   connectionid       : /dmo/connection_id;
15   flightdate         : /dmo/flight_date;
16   @Semantics.amount.currencyCode : 'zrap_dbook_1234.currencycode'
17   flightprice        : /dmo/flight_price;
18   currencycode       : /dmo/currency_code;
19   createdby          : syuname;
20   lastchangedby      : syuname;
21   locallastchangedat : timestampl;
22   "%admin"           : include sych_bdl_draft_admin_inc;
23
24 }
```

10. Enregistrez , activez et fermez la table.
    Certains avertissements s'affichent dans la vue *Problème*. Vous allez maintenant les supprimer.

Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Console | Debug

0 errors, 3 warnings, 0 others

Description	Resource
Warnings (3 items)	
If "with draft" is used, flag a "total etag" field.	zi_rap_travel_1234.as
In the case of "with draft", a composition ("_BOOKING") that does not point to a self lockable entity ("lock master") is given the property "with draft" as a mandatory property.	zi_rap_travel_1234.as
In the case of "with draft", an association ("_TRAVEL") that is "to parent" or for which "dependent by" is used must also have the property "with draft".	zi_rap_travel_1234.as

11. Remplacez la définition d'association dans la définition de comportement de base pour résoudre les avertissements indiquant que les associations sont implicitement activées en mode brouillon car il s'agit d'un objet métier activé en mode brouillon.

Pour cela, utilisez l'extrait de code fourni ci-dessous pour remplacer celui actuellement défini dans la définition du comportement *de déplacement* comme indiqué sur la capture d'écran.

```
association _Booking { create; with draft; }
```

```
*ZI_RAP_TRAVEL_1234
managed;
with draft;

define behavior for ZI_RAP_Travel_1234 alias Travel
implementation in class zbp_i_rap_travel_1234 unique
  update;
  delete;
  association _Booking { create; with draft; }

  field ( numbering : managed, readonly ) TravelUUID;
```

12. Faites de même pour l'association **_Travel** de l' entité *de réservation* .

Pour cela, utilisez l'extrait de code fourni ci-dessous pour remplacer celui actuellement défini dans la définition du comportement *de réservation* comme indiqué sur la capture d'écran.

```
association _Travel { with draft; }
```

```
*ZI_RAP_TRAVEL_1234
52    }
53 }
54
55 define behavior for ZI_RAP_Booking_1234 alias Booking
56 implementation in class zbp_i_rap_booking_1234 unique
57 persistent table zrap_abook_1234
58 draft table zrap_dbook_1234
59 lock dependent by _Travel
60 authorization dependent by _Travel
61 etag master LocalLastChangedAt
62 {
63   association _Travel { with draft; }
64
65   update;
66   delete;
```

13. Spécifiez un *champ « etag total »* dans l'entité *racine* de votre objet BO. Ce champ est nécessaire pour identifier les modifications apportées aux instances *actives* lorsque le verrou durable a expiré. Ce champ **LastChangedA t**sera utilisé à cette fin dans le présent scénario.

Pour cela, utilisez l'extrait de code fourni ci-dessous pour remplacer celui actuellement défini dans la définition du comportement *de déplacement* comme indiqué sur la capture d'écran.

```
lock master total etag LastChangedAt
```

```
*ZI_RAP_TRAVEL_1234
1  managed;
2  with draft;
3
4  define behavior for ZI_RAP_Travel_1234 alias Travel
5  implementation in class zbp_i_rap_travel_1234 unique
6  persistent table zrap_atrav_1234
7  draft table zrap_dtrav_1234
8  lock master total etag LastChangedAt
9  authorization master ( instance )
10 etag master LocalLastChangedAt
11 {
12   create;
```

14. Lorsqu'une instance de brouillon est sur le point d'être activée, l'interface utilisateur des éléments SAP Fiori appelle l'action *de détermination de brouillon* **prepare** dans le backend. Cet appel s'effectue dans un *ensemble de modifications OData* distinct afin de permettre l'enregistrement des messages d'état, même en cas d'échec de l'activation suite à des validations.
    Pour exécuter les validations lors de la préparation, vous devez les affecter au **draft determine action prepare** déclencheur.

Pour cela, insérez l'extrait de code fourni ci-dessous dans la définition du comportement *de déplacement* comme indiqué sur la capture d'écran.

```
draft determine action Prepare  {

   validation validateAgency;

   validation validateCustomer;

   validation validateDates;

 }
```

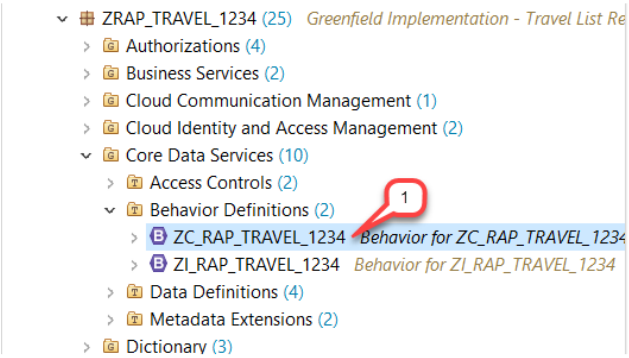
```
*ZI_RAP_TRAVEL_1234
25
26  determination InitialStatus on modify { create; }
30  validation validateAgency on save { field AgencyID; create; }
31  validation validateCustomer on save { field CustomerID; create; }
32  validation validateDates on save { field BeginDate, EndDate; create; }
33
34  draft determine action Prepare
35  {
36    validation validateAgency;
37    validation validateCustomer;
38    validation validateDates;
39  }
40
41  mapping for zrap_atrav_1234
```

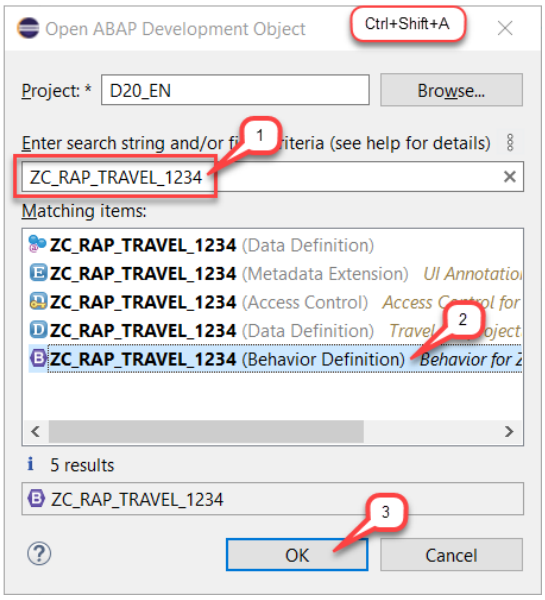
15. Enregistrez et activez les modifications.

Étape 2 : Activer la gestion des brouillons dans l'objet métier projeté

Afin d'activer la gestion des brouillons dans votre application de voyage, vous devez exposer la fonctionnalité dans la couche de projection d'objet métier, c'est-à-dire dans la projection de définition de comportement.

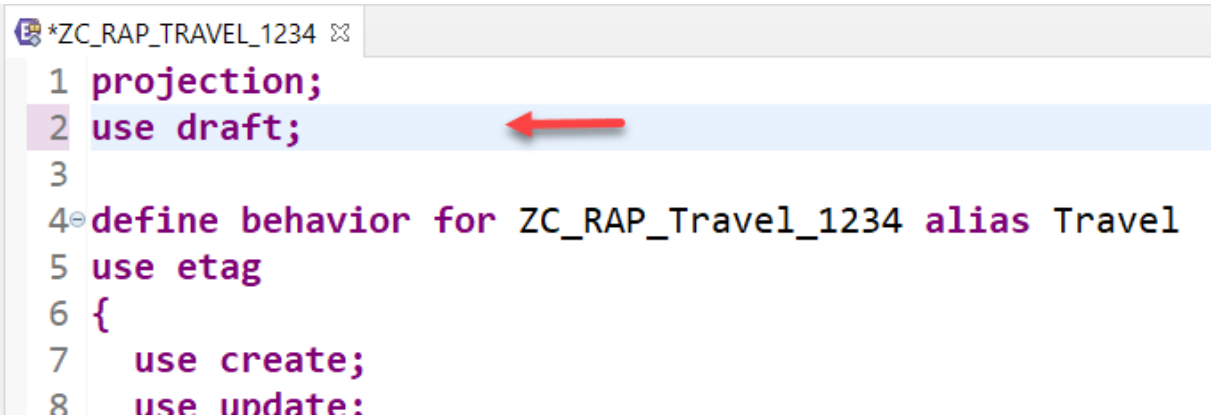
1. Ouvrez la *projection de définition de comportement* **ZC_RAP_Travel_####** de votre objet métier en double-cliquant dessus dans l'Explorateur *de projets* ou en utilisant le raccourci **Ctrl+Maj+A** (*Ouvrir l'objet de développement ABAP*).





2. Activez la gestion des brouillons dans la projection en ajoutant l'instruction **use draft;** dans la section d'en-tête.
   Sinon, la projection se comportera comme si aucun brouillon n'avait été activé pour l'objet métier.

```
use draft;
```



3. Activez la gestion des brouillons pour les associations exposées dans la projection. Pour cela, utilisez les extraits de code ci-dessous pour remplacer les instructions correspondantes dans la projection de définition de comportement, comme illustré dans la capture d'écran.

Extrait de code pour l'entité *de voyage* :

```
  use association _Booking { create; with draft; }
```

Extrait de code pour l'entité *de réservation* :

```
use association _Travel { with draft; }
```

ZC_RAP_TRAVEL_1234

```
1  projection;
2  use draft;
3
4  define behavior for ZC_RAP_Travel_1234 alias Travel
5  use etag
6  {
7    use create;
8    use update;
9    use delete;
10
11   use association _Booking { create; with draft; }
12
13   use action acceptTravel;
14   use action rejectTravel;
15 }
16
17 define behavior for ZC_RAP_Booking_1234 alias Booking
18 use etag
19 {
20   use update;
21   use delete;
22
23   use association _Travel { with draft; }
24 }
```

4. **Restriction pour OData V2 :**
   étant donné que les interfaces utilisateur basées sur des éléments SAP Fiori ne prennent pas actuellement en charge la gestion *Etag* ainsi que la gestion des brouillons pour *OData V2* , vous devez la désactiver dans la définition du comportement projeté.

Pour cela, il suffit de commenter (en utilisant **//**) la ligne d'instruction **use etag** dans la projection.

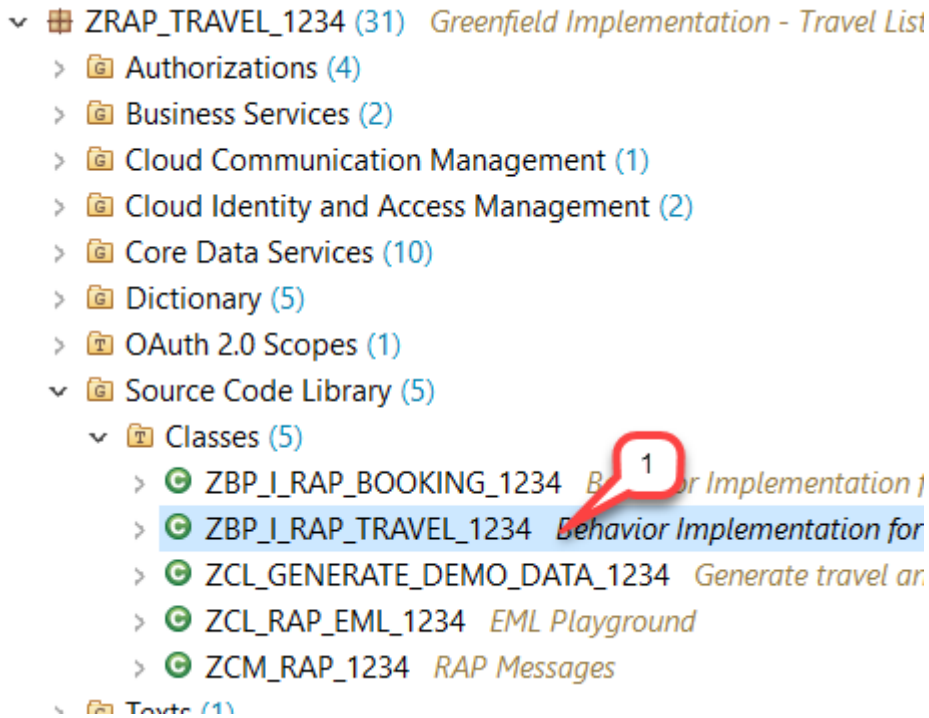
```
*ZC_RAP_TRAVEL_1234
1  projection;
2  use draft;
3
4  define behavior for ZC_RAP_Travel_1234 alias Travel
5  //use etag
6  {
7    use create;
8    use update;
14   use action rejectTravel;
15 }
16
17 define behavior for ZC_RAP_Booking_1234 alias Booking
18 //use etag
19 {
20   use update;
21   use delete;
```

5. Enregistrez et activez la projection de définition de comportement.

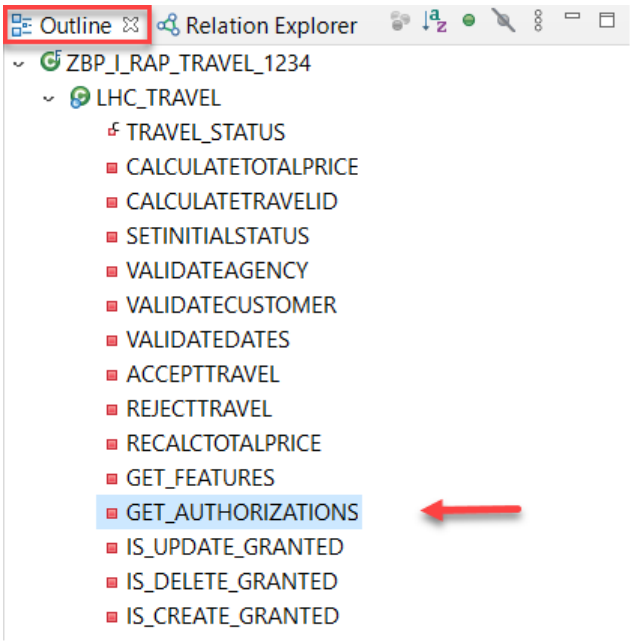
Étape 3 : Ajuster l'implémentation du comportement de l'objet métier

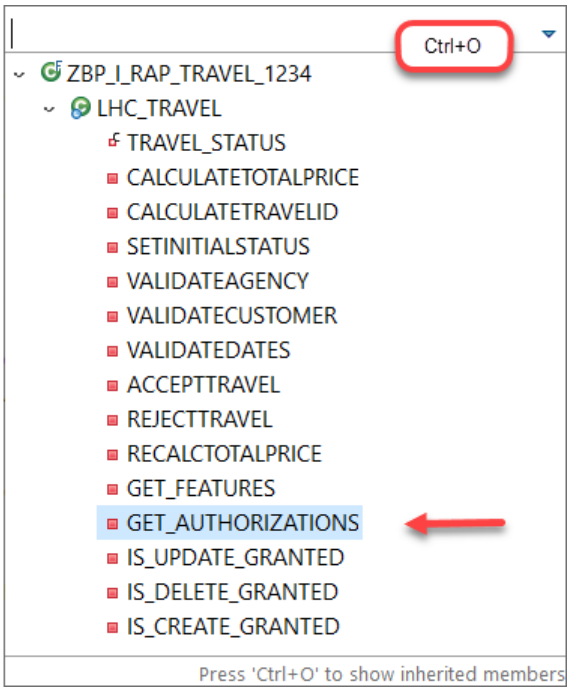
Vous allez maintenant effectuer quelques ajustements dans la mise en œuvre du comportement de votre comportement de voyage pour permettre la prise en compte des instances de brouillon.

1. Ouvrez la classe d'implémentation du comportement **ZBP_I_RAP_TRAVEL_####** de l'entité de voyage dans l'Explorateur *de projets*



2. Accédez à l'implémentation de la méthode **get_authorizations**.
   Vous pouvez utiliser la vue *Plan* ou le raccourci ADT **Ctrl+O** pour cela.





3. L'introduction de la gestion *des brouillons* a ajouté deux actions prédéfinies à l'objet métier : l' *action de détermination des brouillons* **Prepare** et l'action *de brouillon* **Edit** . Faites défiler vers le bas et supprimez les commentaires des lignes de code des actions **Prepare** et **Edit** du **COND** constructeur, comme illustré dans les captures d'écran ci-dessous.

Cette opération doit être effectuée à deux endroits dans le code source. Les deux méthodes sont désormais également prises en compte dans le *contrôle d'autorisation.*

Première section :

```
*ZBP_I_RAP_TRAVEL_1234
ZBP_I_RAP_TRAVEL_1234 ▸ LHC_TRAVEL ▸ GET_AUTHORIZATIONS
403     SELECT FROM zrap_atrav_1234
404       FIELDS travel_uuid,overall_status
405       FOR ALL ENTRIES IN @travels
406       WHERE travel_uuid EQ @travels-TravelUUID
407       ORDER BY PRIMARY KEY
408       INTO TABLE @DATA(travels_before_image).
409
410     is_update_requested = COND #( WHEN requested_authorizations-%update              = if_abap_behv=>mk-on OR
411                                   requested_authorizations-%action-acceptTravel = if_abap_behv=>mk-on OR
412                                   requested_authorizations-%action-rejectTravel = if_abap_behv=>mk-on OR
413                                   requested_authorizations-%action-Prepare      = if_abap_behv=>mk-on OR
414                                   requested_authorizations-%action-Edit         = if_abap_behv=>mk-on OR
415                                   requested_authorizations-%assoc-_Booking      = if_abap_behv=>mk-on
416                                 THEN abap_true ELSE abap_false ).
417
418     is_delete_requested = COND #( WHEN requested_authorizations-%delete = if_abap_behv=>mk-on
419                                   THEN abap_true ELSE abap_false ).
420
```

Deuxième partie :

```
*ZBP_I_RAP_TRAVEL_1234
ZBP_I_RAP_TRAVEL_1234 ▸ LHC_TRAVEL ▸ GET_AUTHORIZATIONS ▸ loop
456                     ) TO reported-travel.
457         ENDIF.
458       ENDIF.
459
460       APPEND VALUE #( %tky = travel-%tky
461
462                       %update              = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
463                       %action-acceptTravel = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
464                       %action-rejectTravel = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
465                       %action-Prepare      = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
466                       %action-Edit         = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
467                       %assoc-_Booking      = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
468
469                       %delete              = COND #( WHEN delete_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
470                     )
471         TO result.
472     ENDLOOP.
473   ENDMETHOD.
```

4. Enregistrez et activez l'implémentation du comportement.

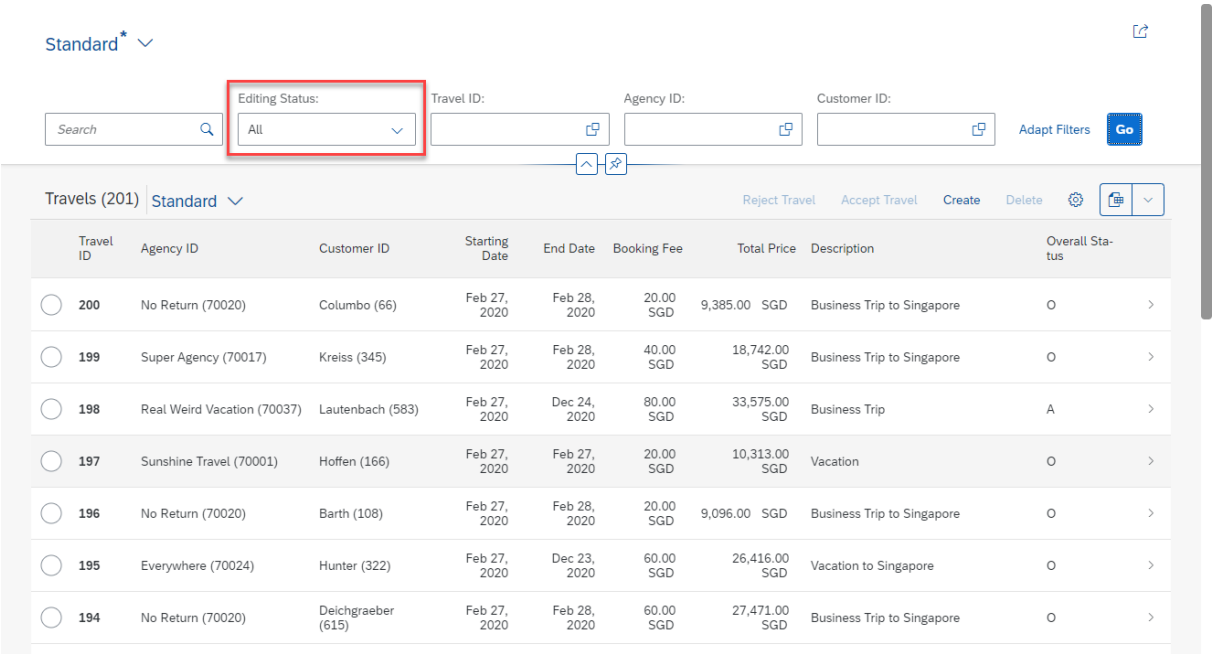
Étape 4 : Prévisualisez l'application Voyage améliorée et testez-la

Vous avez maintenant terminé les implémentations de comportement et pouvez exécuter et vérifier les éléments améliorés de l'application de voyage SAP Fiori.

1. Lancez l'application *Voyage* dans votre liaison de service **ZUI_RAP_TRAVEL_O2_####** ou actualisez- F5la ( ) dans le navigateur. Fournissez vos identifiants ABAP si nécessaire.

Appuyez sur **Go** sur l'interface utilisateur pour charger les données du back-end.

La première chose que vous remarquerez est un nouveau champ de filtre qui permet de filtrer sur l'état *d'édition* qui est désormais affiché dans la zone de filtre.

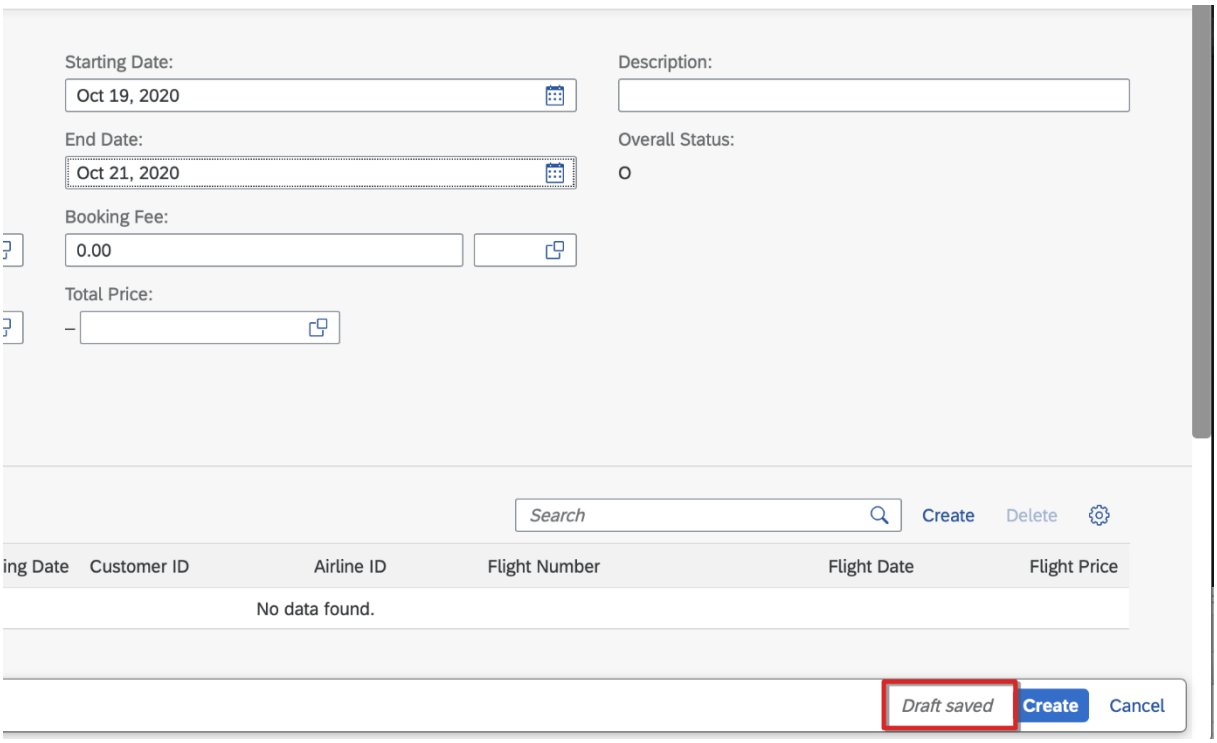


2. N'hésitez pas à tester votre application.

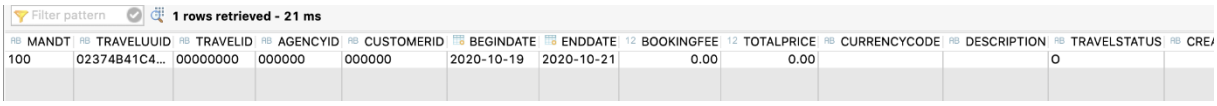
Par exemple, créez un nouveau voyage.

Contrairement au cas d'utilisation sans brouillon, la détermination *lors de la modification* pour la mise en défaut du *statut* a déjà été exécutée.

Lors de la modification des champs sur la page de l'objet, vous voyez immédiatement l'indicateur *d'enregistrement du brouillon* en bas.



La vérification du tableau de brouillons associé dans le backend vous montre les valeurs saisies jusqu'à présent.



Filter pattern | 1 rows retrieved - 21 ms

MANDT	TRAVELUUID	TRAVELID	AGENCYID	CUSTOMERID	BEGINDATE	ENDDATE	BOOKINGFEE	TOTALPRICE	CURRENCYCODE	DESCRIPTION	TRAVELSTATUS	CREA
100	02374B41C4...	00000000	000000	000000	2020-10-19	2020-10-21	0.00	0.00			O	

Fournissez une valeur incorrecte pour l'ID d'agence (**AgencyID**) et des valeurs correctes pour l'ID client (**CustomerID**) ainsi que pour les **BeginDate** dates de début et de fin (EndDate).

Choisissez **Enregistrer**.

Vous recevrez un message correspondant du backend.
S'agissant d'un *message d'état,* il est stocké dans la base de données et appartient à l'état de l'instance de l'objet métier.

Vous pouvez quitter l'écran, filtrer pour votre nouvel enregistrement *de voyage* et vous verrez les messages directement lors de l'ouverture de la page de l'objet, sans revalider l'instance sur le backend.

Corrigez la valeur et choisissez **Enregistrer**.

Maintenant, la détermination **onSave** est effectuée qui calcule le **TravelID**.

Sans avoir besoin de sauvegarder, vous pouvez modifier l'en-tête *du voyage* ainsi que les entrées *de réservation*.

Résumé

Dans cette unité, vous avez appris à activer la gestion des brouillons dans votre objet métier Voyage.

5. Semaine 4 — *Intégration de Code Métier Existant (Unmanaged Runtime)*

L'objectif de cette semaine est d'apprendre à **intégrer du code métier existant** (appelé approche *Brownfield*) dans une application Fiori Elements basée sur RAP, en utilisant le **mode unmanaged**. Cela permet de **réutiliser la logique métier déjà existante** (fonction modules, structures, tables, etc.) tout en tirant parti de l'architecture moderne RAP.

Concrètement, vous allez:

- Étudier le **code métier existant** dans le package /DMO/FLIGHT_LEGACY (tables /DMO/TRAVEL, /DMO/BOOKING, fonctions /DMO/FLIGHT_TRAVEL_CREATE, etc.).
- Créer une **application Fiori Elements** qui exploite ce code via une **implémentation unmanaged**.
- Définir le **modèle de données CDS**, le **comportement BO personnalisé**, la **projection RAP** et l'**exposition via un service OData UI**.
- Comprendre comment **EML** (Entity Manipulation Language) interagit avec le buffer métier via les **modules de fonction**.

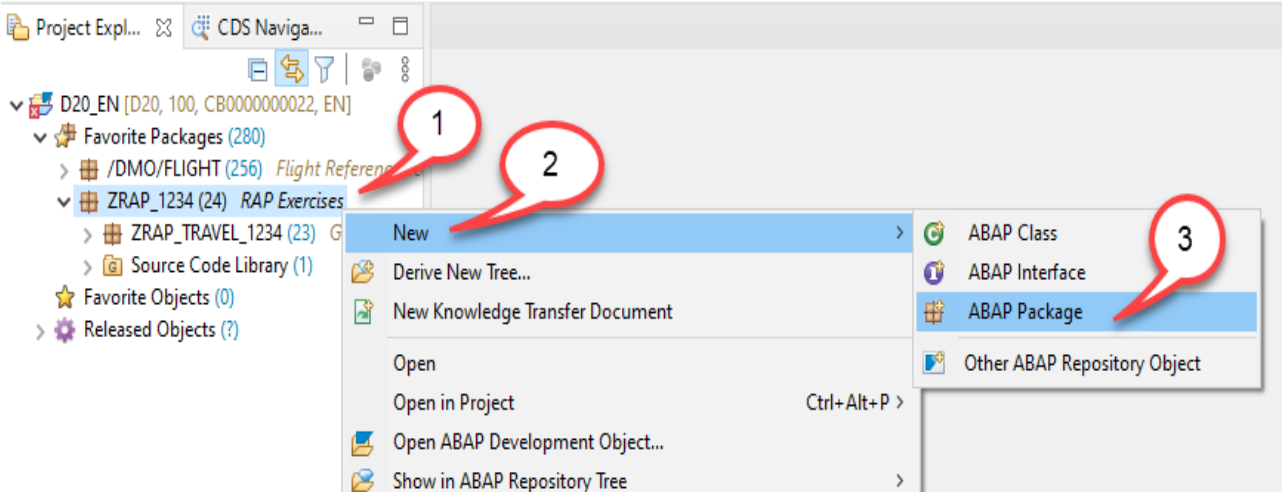
5.1 Création du modèle de données CDS (Unité 1)

Objectif de l'unité

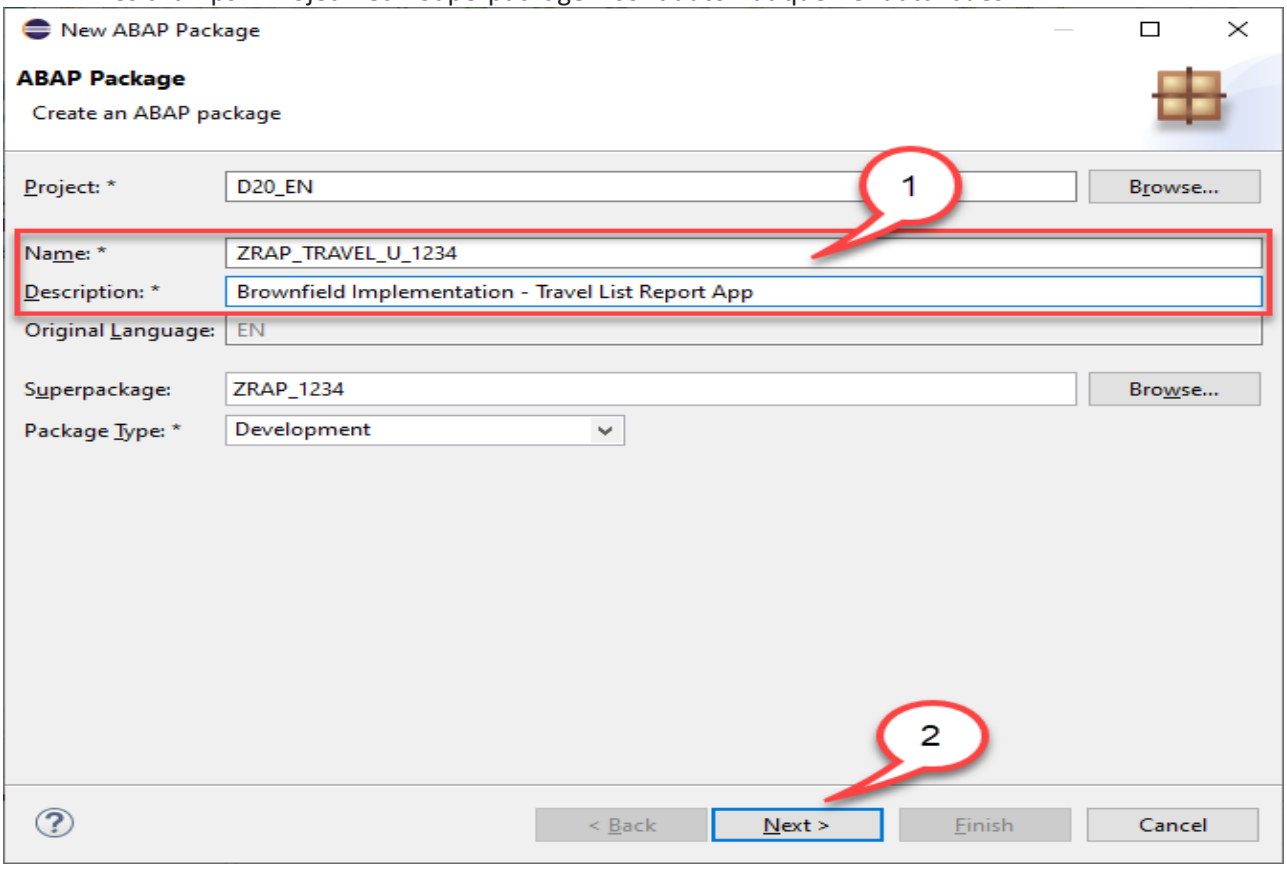
Définir les **vues CDS racine et enfant** à partir des tables legacy /DMO/TRAVEL et /DMO/BOOKING pour construire le **modèle de données** de l'application RAP en mode **unmanaged**.

Étape 1 : Créer le package

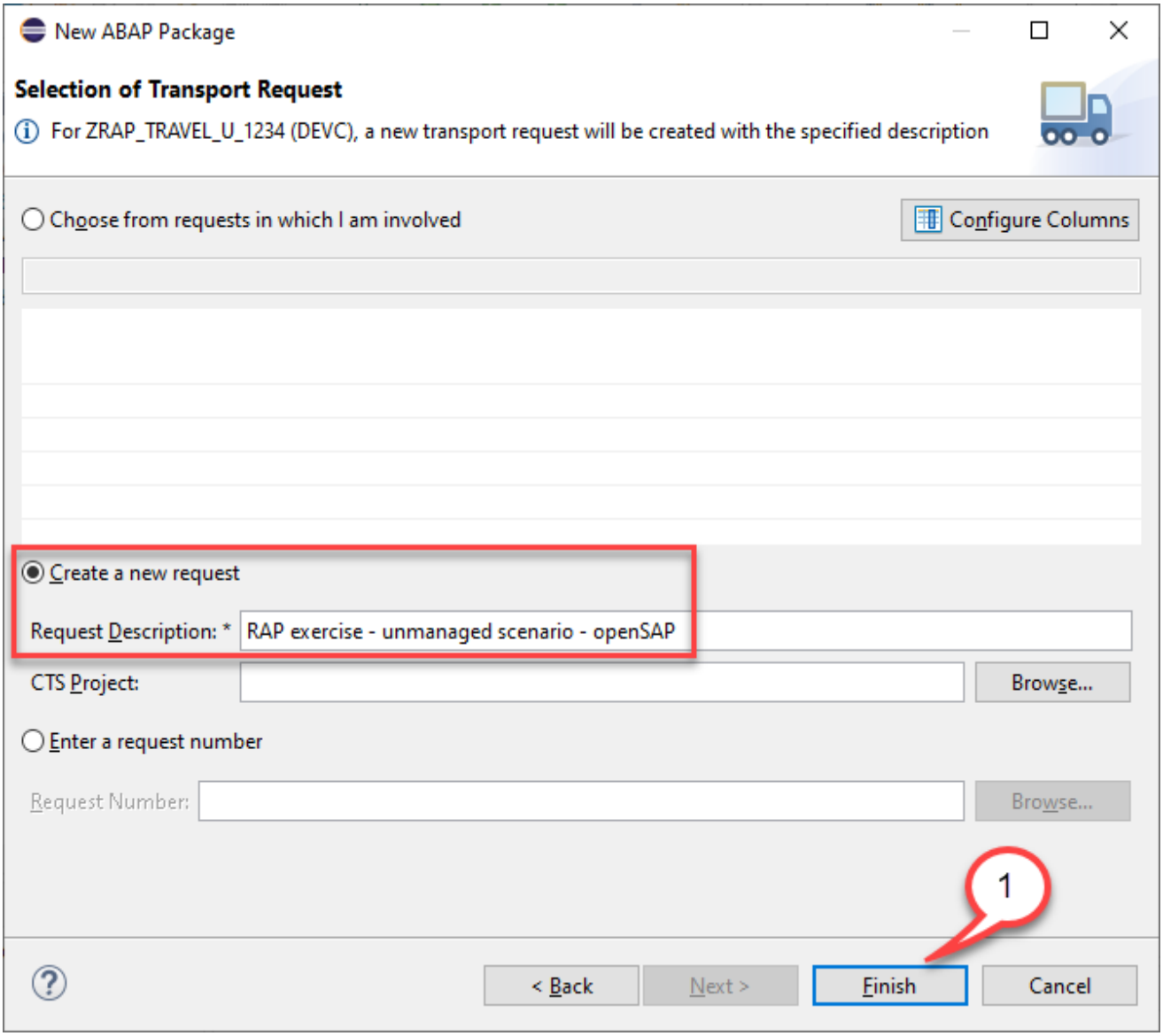
1. Créez d'abord un nouveau package pour regrouper les différents artefacts de développement que vous allez créer lors de l'implémentation de votre application de voyage. Il s'agira d'un sous-package du package **ZRAP_####** créé lors de la semaine 1.



2. Conservez **ZRAP_TRAVEL_U_####** le nom (où _U indique qu'il s'agit d'un scénario non géré) et une description pertinente (par exemple, « Implémentation Brownfield - Application de rapport de liste de déplacements »), puis cliquez sur « **Suivant** » pour continuer.
Les champs « Projet » et « Superpackage » sont automatiquement attribués.



4. Sélectionnez une demande de transport existante ou créez-en une nouvelle, puis choisissez **Terminer** pour créer le nouveau package.

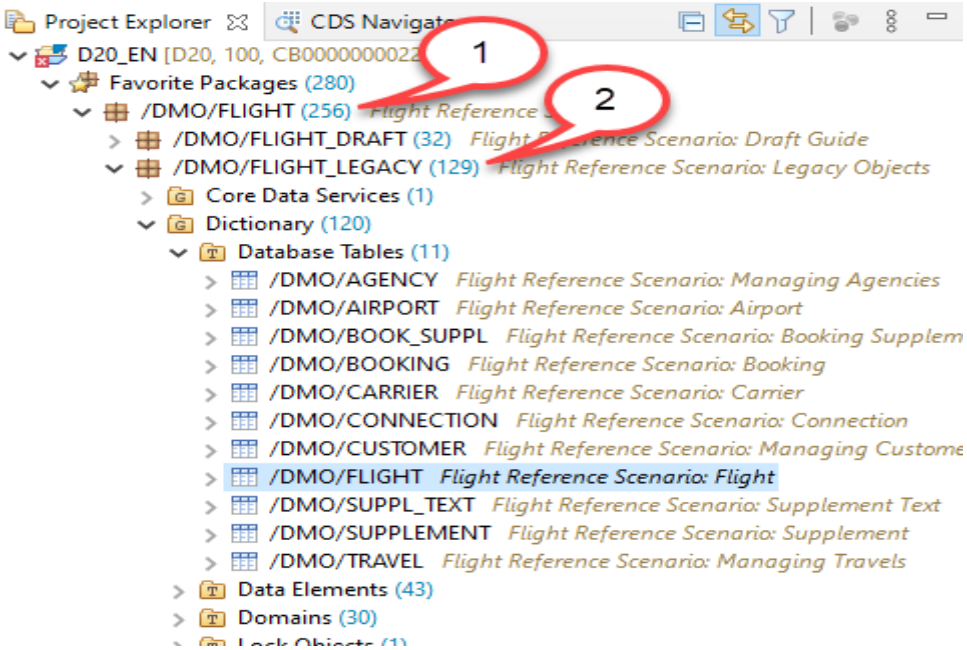


Le package est maintenant créé.

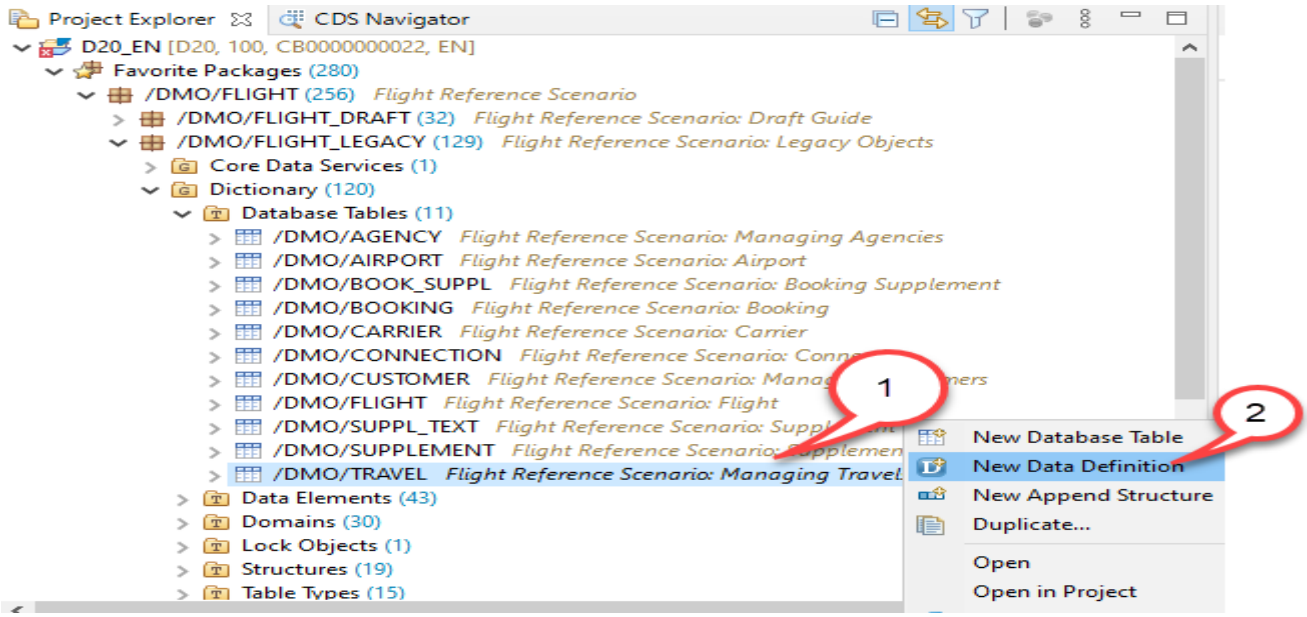
Étape 2 : Créer une entité de vue racine – Voyage

Commençons par créer notre interface racine pour les données de voyage. Nous utiliserons le package **ZRAP_TRAVEL_U_####** que vous avez créé précédemment.
Pour créer une définition de données pour une vue CDS, un nouvel assistant est disponible. Il vous permet de créer une définition de données depuis le menu contextuel d'une table ou d'une vue CDS.
Pour utiliser cet assistant, procédez comme suit :

1. Dans le **Favorites Packages**dossier, développez le package **/DMO/FLIGHT** puis le sous-package **/DMO/FLIGHT_LEGACY**. Développez le dossier Dictionary puis le dossier Database Tables.

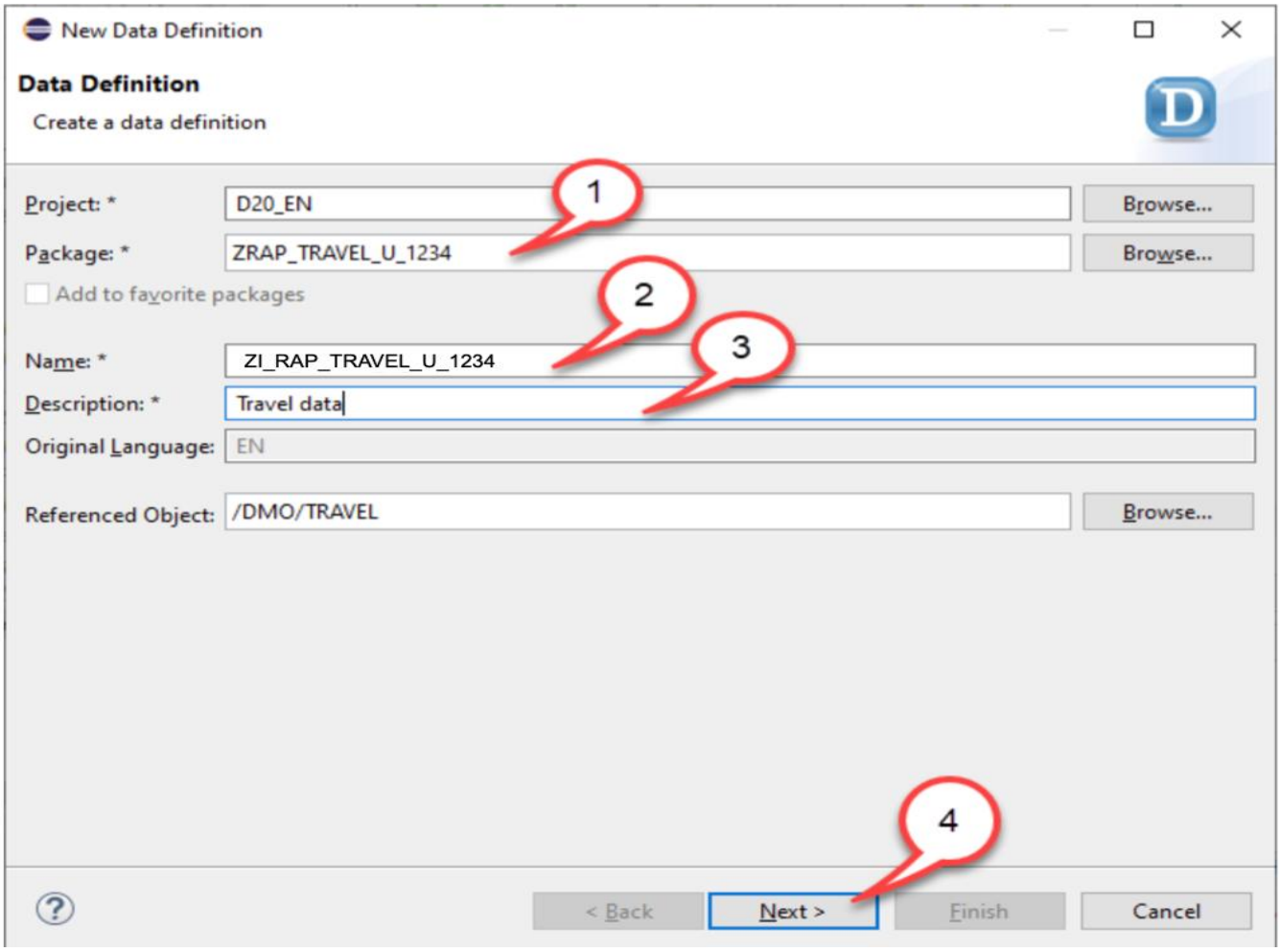


2. Faites un clic droit sur le tableau **/DMO/Travel** et sélectionnez **New Data Definition**



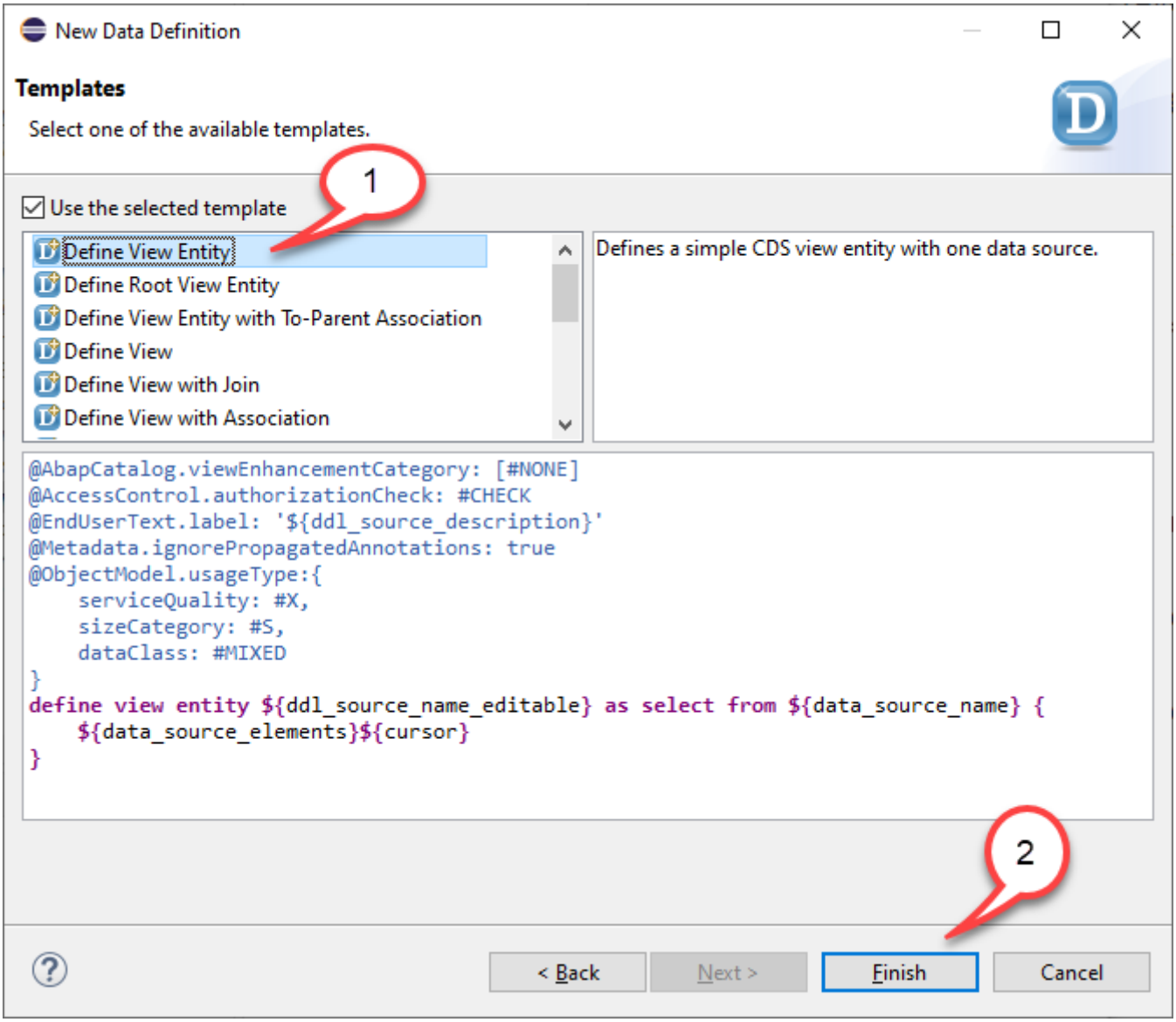
3. Cela lancera le nouvel assistant d'ADT, qui vous permettra de créer facilement une vue CDS basée sur une table ou une autre vue CDS. Saisissez les valeurs suivantes.

- Nom du package : **ZRAP_TRAVEL_U_####**
- Nom : **ZI_RAP_TRAVEL_U_####**
- Description : **Travel data**



4. Sélectionnez une demande de transport existante ou créez-en une nouvelle, puis choisissez **Suivant**.

5. Dans la boîte de dialogue **Nouvelle définition de données,** choisissez le modèle **Définir l'entité de vue** et appuyez sur **Terminer** pour créer l'entité de vue racine.



Nous commençons à éditer le modèle en ajoutant d'abord le mot clé **root**.

```
define root view entity ZI_RAP_Travel_U_#### as select from /DMO/TRAVEL {

    ///DMO/TRAVEL

    key travel_id,

....
```

Et nous ajoutons les associations suivantes

```
association [0..1] to /DMO/I_Agency   as _Agency   on $projection.AgencyID =
_Agency.AgencyID

association [0..1] to /DMO/I_Customer as _Customer on $projection.CustomerID =
_Customer.CustomerID

association [0..1] to I_Currency      as _Currency on $projection.CurrencyCode =
_Currency.Currency
```

Afin d'obtenir des noms de champs clairs, nous ajouterons des pour les noms de champs de base de données. Cela signifie que nous ajoutons un alias **TravelID quand** le nom du champ dans la base de données est **travel_id**.

```
{

    key travel_id     as TravelID,

        agency_id     as AgencyID,

      ....

  }
```

7. Remplacez le code de définition des données de voyage dans l'éditeur par l'extrait de code ci-dessous.

   Enregistrez les modifications, mais **n'activez PAS** encore la vue de l'interface de voyage.

```
@EndUserText.label : « Données de voyage »



définir l'entité de vue racine ZI_RAP_Travel_U_####

 comme sélectionner dans /dmo/travel comme Voyage



 association [0..1] à /DMO/I_Agency comme _Agency sur $projection.AgencyID = _Agency.AgencyID

 association [0..1] à /DMO/I_Customer en tant que _Customer sur $projection.CustomerID =
_Customer.CustomerID

 association [0..1] à I_Currency comme _Currency sur $projection.CurrencyCode =
_Currency.Currency
```

```
{
 clé travel_id comme TravelID,
    agency_id comme AgencyID,
    customer_id comme CustomerID,
    begin_date comme BeginDate,
    end_date comme EndDate,
    @Semantics.amount.currencyCode : « CurrencyCode »
    booking_fee comme BookingFee,
    @Semantics.amount.currencyCode : « CurrencyCode »
    total_price comme PrixTotal,
    currency_code comme CurrencyCode,
    description comme Description,
    statut comme Statut,
    @Semantics.user.createdBy : vrai
    créé par comme Créé par,
    @Semantics.systemDateTime.createdAt : vrai
    créé à comme Createdat,
    @Semantics.user.lastChangedBy : vrai
    lastchangedby comme Lastchangedby,
    @Semantics.systemDateTime.lastChangedAt : vrai
    lastchangedat comme Lastchangedat,

    /* associations */
```

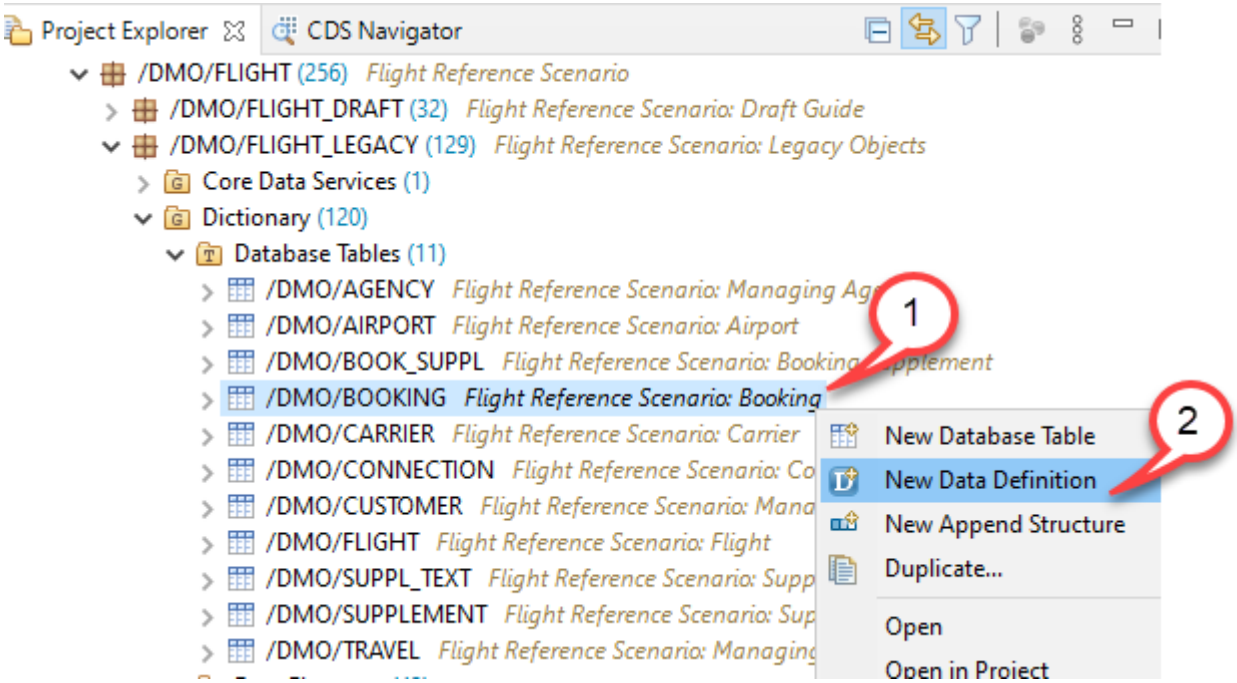
```
   _Agence,

   _Client,

   _Devise

}@EndUserText.label: 'Travel data'



define root view entity ZI_RAP_Travel_U_####

  as select from /dmo/travel as Travel



  association [0..1] to /DMO/I_Agency         as _Agency   on $projection.AgencyID
= _Agency.AgencyID

  association [0..1] to /DMO/I_Customer       as _Customer on
$projection.CustomerID = _Customer.CustomerID

  association [0..1] to I_Currency            as _Currency on
$projection.CurrencyCode = _Currency.Currency



{

  key travel_id     as TravelID,

      agency_id     as AgencyID,

      customer_id   as CustomerID,

      begin_date    as BeginDate,

      end_date      as EndDate,

      @Semantics.amount.currencyCode: 'CurrencyCode'

      booking_fee   as BookingFee,

      @Semantics.amount.currencyCode: 'CurrencyCode'

      total_price   as TotalPrice,

      currency_code as CurrencyCode,

      description   as Description,
```

```
      status        as Status,
      @Semantics.user.createdBy: true
      createdby     as Createdby,
      @Semantics.systemDateTime.createdAt: true
      createdat     as Createdat,
      @Semantics.user.lastChangedBy: true
      lastchangedby as Lastchangedby,
      @Semantics.systemDateTime.lastChangedAt: true
      lastchangedat as Lastchangedat,

      /* associations */

      _Agency,
      _Customer,
      _Currency
}
```

Étape 3 : Créer une entité de vue pour les données de réservation

Comme cela a été fait précédemment avec la table de base de données de voyage, nous allons créer une entité de vue pour les données de réservation.

1. Faites un clic droit sur le tableau **/DMO/Booking** et sélectionnez **New Data Definition**



2. Cela lancera le nouvel assistant d'ADT, qui vous permettra de créer facilement une vue CDS basée sur une table ou une autre vue CDS. Saisissez les valeurs suivantes.

- Nom du package : **ZRAP_TRAVEL_U_####**
- Nom : **ZI_RAP_BOOKING_U_####**
- Description : **Booking data**

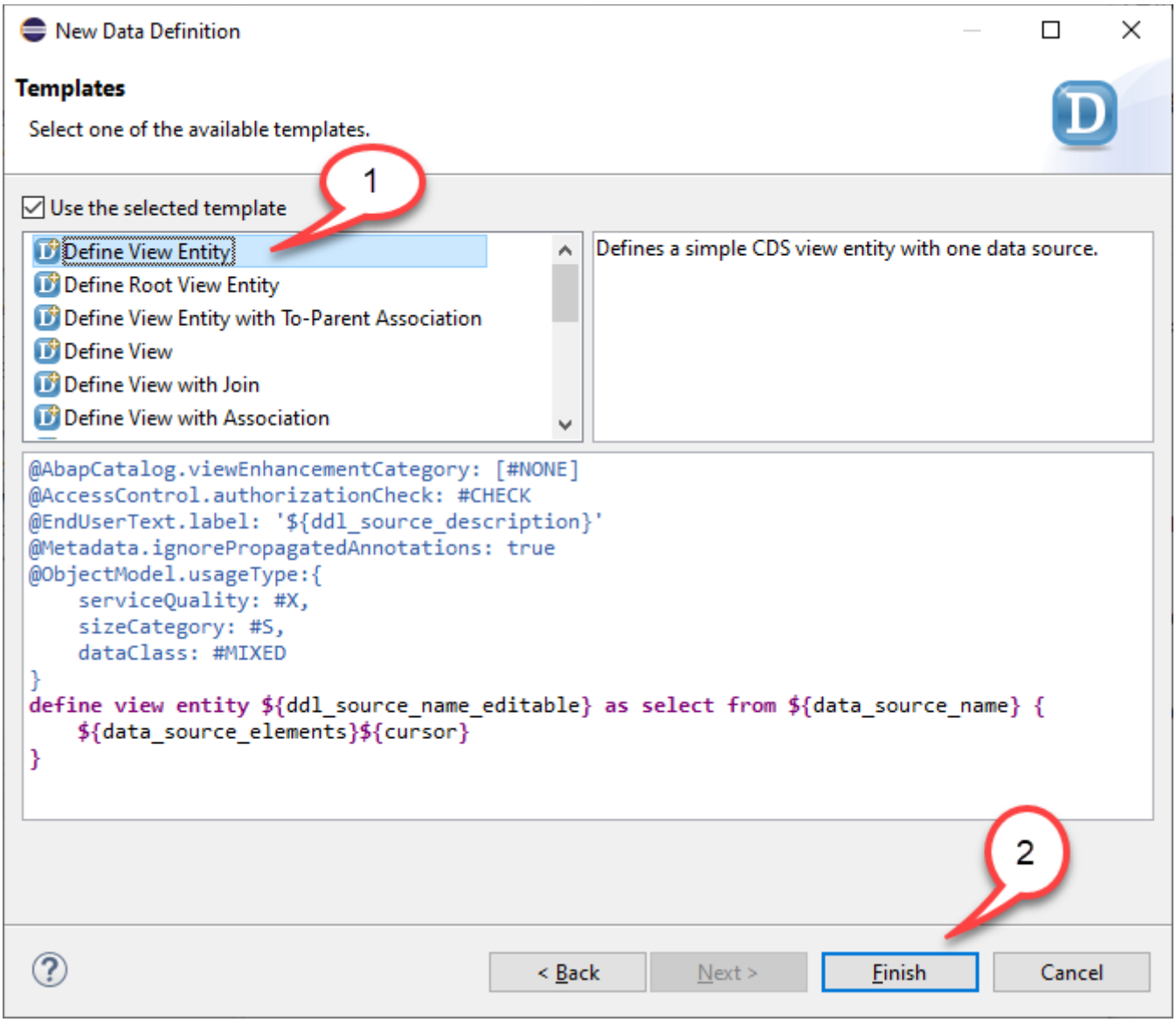
Le nom de l'objet référencé sera **/DMO/Booking**. Appuyez sur **Suivant**.

Nous suivons à nouveau la convention de dénomination du modèle de données VDM de SAP S/4HANA.

- L'espace de noms**Z**
- Le préfixe **I_**car il s'agit d'une vue d'interface
- Le préfixe que **RAP_**nous utilisons simplement pour ce cours
- **Booking**est le nom de l'entité
- Le suffixe **_U**indique que nous utilisons une implémentation non gérée
- Le suffixe **_####**que nous utilisons simplement pour ce cours pour indiquer votre numéro de groupe.

4. Sélectionnez une demande de transport existante ou créez-en une nouvelle, puis choisissez **Suivant**.

5. Dans la boîte de dialogue **Nouvelle définition de données,** choisissez le modèle **Définir l'entité de vue** et appuyez sur **Terminer** pour créer l'entité de vue racine.



Nous commençons à éditer le modèle en ajoutant les associations suivantes

```
 association [1..1] to /DMO/I_Carrier              as _Carrier    on
$projection.CarrierID = _Carrier.AirlineID

  association [1..1] to /DMO/I_Customer             as _Customer   on
$projection.CustomerID = _Customer.CustomerID

  association [1..1] to /DMO/I_Connection           as _Connection on
$projection.CarrierID    = _Connection.AirlineID
```

```
                                                                    and
$projection.ConnectionID = _Connection.ConnectionID

  association [1..1] to /DMO/I_Flight              as _Flight     on
$projection.CarrierID    = _Flight.AirlineID

                                                                    and
$projection.ConnectionID = _Flight.ConnectionID

                                                                    and
$projection.FlightDate   = _Flight.FlightDate
```

Ensuite, nous modifions le code source DDL entre les accolades afin que tous les champs obtiennent de jolis alias et que le champ flight_price obtienne en plus l'annotation **@Semantics.amount.currencyCode: 'CurrencyCode'**qui pointe vers le champ **CurrencyCode**qui contient les informations sur le code de devise.

7. Remplacez le code de définition des données de voyage dans l'éditeur par l'extrait de code ci-dessous.

Enregistrez les modifications, mais **n'activez PAS** encore la vue de l'interface de voyage.

```
@EndUserText.label: 'Booking data'

@Metadata.ignorePropagatedAnnotations: true



define view entity ZI_RAP_Booking_U_####

  as select from /dmo/booking





  association [1..1] to /DMO/I_Carrier    as _Carrier    on  $projection.CarrierID
= _Carrier.AirlineID

  association [1..1] to /DMO/I_Customer   as _Customer   on
$projection.CustomerID = _Customer.CustomerID

  association [1..1] to /DMO/I_Connection as _Connection on  $projection.CarrierID
= _Connection.AirlineID

                                                          and
$projection.ConnectionID = _Connection.ConnectionID
```

```
  association [1..1] to /DMO/I_Flight     as _Flight     on  $projection.CarrierID
= _Flight.AirlineID

                                                         and
$projection.ConnectionID = _Flight.ConnectionID

                                                         and
$projection.FlightDate   = _Flight.FlightDate


{

  key travel_id     as TravelID,

  key booking_id    as BookingID,

      booking_date  as BookingDate,

      customer_id   as CustomerID,

      carrier_id    as CarrierID,

      connection_id as ConnectionID,

      flight_date   as FlightDate,

      @Semantics.amount.currencyCode: 'CurrencyCode'

      flight_price  as FlightPrice,

      currency_code as CurrencyCode,


      /* associations */


      _Carrier,

      _Customer,

      _Connection,

      _Flight


}
```

Étape 4 : Activez vos modifications

Vous pouvez désormais activer vos modifications en appuyant sur le bouton « Activer les objets de développement ABAP inactifs » ou en appuyant sur « Ctrl+Maj+F3 ».

Étape 5 : Vérifiez l'aperçu des données

Vous pouvez désormais vérifier que les données sont récupérées pour les deux entités, les données de voyage ainsi que les données de réservation.

[D20] ZRAP_TRAVEL_U_1234 | [D20] ZI_RAP_TRAVEL_U_1234 | [D20] ZI_RAP_BOOKING_U_1234 | [D20] ZI_RAP_TRAVEL_U_1234

ZI_RAP_TRAVEL_U_1234

Raw Data | Show Log | Max. Rows: 100

Filter pattern | 100 rows retrieved - 13 ms (partial result) | SQL Console | Number of Entries | Select Columns | Add filter

TravelID	AgencyID	CustomerID	BeginDate	EndDate	BookingFee	TotalPrice	CurrencyCode
00000001	070041	000594	2020-02-29	2020-12-27	40.00	1889.00	USD
00000002	070007	000608	2020-02-29	2020-02-29	20.00	900.00	USD
00000003	070046	000093	2020-02-29	2020-12-27	80.00	4164.00	USD
00000004	070042	000665	2020-02-29	2020-12-27	40.00	1871.00	USD
00000005	070007	000225	2020-02-29	2020-02-29	20.00	992.00	USD
00000006	070049	000072	2020-02-29	2020-12-27	120.00	5586.00	USD
00000007	070046	000138	2020-02-29	2020-12-27	120.00	5691.00	USD
00000008	070012	000705	2020-02-29	2020-03-02	60.00	2777.00	USD
00000009	070032	000115	2020-02-29	2020-12-27	120.00	5792.00	USD
00000010	070016	000697	2020-02-29	2020-03-02	20.00	959.00	USD

[D20] ZRAP_TRAVEL_... | [D20] ZI_RAP_TRAVE... | [D20] ZI_RAP_BOOKI... | [D20] ZI_RAP_TRAVE... | [D20] ZI_RAP_BOOKI...

ZI_RAP_BOOKING_U_1234

Raw Data | Show Log | Max. Rows: 100

Filter pattern | 100 rows retrieved - 12 ms (partial result) | SQL Console | Number of Entries | Select Columns | Add filter

TravelID	BookingID	BookingDate	CustomerID	CarrierID	ConnectionID	FlightDate	FlightPr
00000001	0001	2020-02-12	000594	UA	1537	2020-02-29	438.
00000001	0002	2020-02-29	000594	AA	0322	2020-03-02	438.
00000001	0003	2020-12-06	000594	UA	1537	2020-12-25	438.
00000001	0004	2020-12-10	000594	AA	0322	2020-12-27	438.
00000002	0001	2020-02-27	000099	UA	1537	2020-02-29	438.
00000002	0002	2020-02-27	000660	UA	1537	2020-02-29	438.
00000003	0001	2020-02-09	000093	UA	1537	2020-02-29	438.
00000003	0002	2020-02-09	000706	UA	1537	2020-02-29	438.
00000003	0003	2020-02-12	000093	AA	0322	2020-03-02	438.
00000003	0004	2020-02-12	000706	AA	0322	2020-03-02	438.
00000003	0005	2020-12-21	000093	UA	1537	2020-12-25	438.

Étape 6 : Ajouter les associations de composition

Dans la vue de l'interface Travel CDS, nous ajoutons une cardinalité un-à-plusieurs [0..] car il peut y avoir plusieurs réservations relatives à une entité de voyage.

```
  composition [0..*] of ZI_RAP_Booking_U_####  as _Booking

  association [0..1] to /DMO/I_Agency         as _Agency   on $projection.AgencyID
= _Agency.AgencyID

  association [0..1] to /DMO/I_Customer       as _Customer on
$projection.CustomerID = _Customer.CustomerID

  association [0..1] to I_Currency            as _Currency on
$projection.CurrencyCode = _Currency.Currency
```

Ces nouvelles associations doivent également être publiées en les ajoutant à la liste des champs

```
/* associations */

      _Booking,

      _Agency,

      _Customer,

      _Currency
```

!!! N'activez **PAS** vos modifications **!!!**

Dans la vue de l'interface CDS de réservation, nous ajoutons une association qui pointe vers l'entité parent.

```
 association        to parent ZI_RAP_Travel_U_#### as _Travel     on
$projection.TravelID = _Travel.TravelID

  association [1..1] to /DMO/I_Carrier    as _Carrier    on  $projection.CarrierID
= _Carrier.AirlineID

  association [1..1] to /DMO/I_Customer   as _Customer   on
$projection.CustomerID = _Customer.CustomerID

  association [1..1] to /DMO/I_Connection as _Connection on  $projection.CarrierID
= _Connection.AirlineID

                                                         and
$projection.ConnectionID = _Connection.ConnectionID

  association [1..1] to /DMO/I_Flight     as _Flight     on  $projection.CarrierID
= _Flight.AirlineID
```

```
                                                                and
$projection.ConnectionID = _Flight.ConnectionID

                                                                and
$projection.FlightDate   = _Flight.FlightDate
```

C'est ici, dans l'entité enfant, que la cardinalité et la condition sont définies.

Cette association doit également être publiée en l'ajoutant à la liste des champs de la vue de l'interface Booking CDS.

```
  /* associations */

     _Travel,

     _Carrier,

     _Customer,

     _Connection,

     _Flight
```

Le code source DDL devrait maintenant ressembler à ceci :

[D20] ZRAP_TRAVEL_... | [D20] ZI_RAP_TRAVE... | [D20] ZI_RAP_BOOKI... | [D20] ZI_RAP_TRAVE... | [D20] ZI_RAP_BOOKI...

```
1  @EndUserText.label: 'Travel data'
2  @Metadata.ignorePropagatedAnnotations: true
3
4  define root view entity ZI_RAP_Travel_U_1234
5    as select from /dmo/travel as Travel
6
7    composition [0..*] of ZI_RAP_Booking_U_1234 as _Booking
8    association [0..1] to /DMO/I_Agency         as _Agency   on $projection.AgencyID = _Agency.AgencyID
9    association [0..1] to /DMO/I_Customer       as _Customer on $projection.CustomerID = _Customer.CustomerID
10   association [0..1] to I_Currency            as _Currency on $projection.CurrencyCode = _Currency.Currency
11
12 {
13   key travel_id     as TravelID,
14       agency_id     as AgencyID,
15       customer_id   as CustomerID,
16       begin_date    as BeginDate,
17       end_date      as EndDate,
18       @Semantics.amount.currencyCode: 'CurrencyCode'
19       booking_fee   as BookingFee,
20       @Semantics.amount.currencyCode: 'CurrencyCode'
21       total_price   as TotalPrice,
22       currency_code as CurrencyCode,
23       description   as Description,
24       status        as Status,
25       @Semantics.user.createdBy: true
26       createdby     as Createdby,
27       @Semantics.systemDateTime.createdAt: true
28       createdat     as Createdat,
29       @Semantics.user.lastChangedBy: true
30       lastchangedby as Lastchangedby,
31       @Semantics.systemDateTime.lastChangedAt: true
32       lastchangedat as Lastchangedat,
33
34       /* associations */
35
36       _Booking,
37       _Agency,
38       _Customer,
39       _Currency
40 }
```

[D20] ZRAP_TRAVEL_... | [D20] ZI_RAP_TRAVE... | [D20] ZI_RAP_BOOKI... | [D20] ZI_RAP_TRAVE... | [D20] ZI_RAP_BOOKI...

```
1  @EndUserText.label: 'Booking data'
2  @Metadata.ignorePropagatedAnnotations: true
3
4  define view entity ZI_RAP_Booking_U_1234
5    as select from /dmo/booking
6
7    association        to parent ZI_RAP_Travel_U_1234 as  Travel     on  $projection.TravelID =  Travel.TravelID
8    association [1..1] to /DMO/I_Carrier    as _Carrier    on  $projection.CarrierID = _Carrier.AirlineID
9    association [1..1] to /DMO/I_Customer   as _Customer   on  $projection.CustomerID = _Customer.CustomerID
10   association [1..1] to /DMO/I_Connection as _Connection on  $projection.CarrierID    = _Connection.AirlineID
11                                                          and $projection.ConnectionID = _Connection.ConnectionID
12   association [1..1] to /DMO/I_Flight     as _Flight     on  $projection.CarrierID    = _Flight.AirlineID
13                                                          and $projection.ConnectionID = _Flight.ConnectionID
14                                                          and $projection.FlightDate   = _Flight.FlightDate
15
16 {
17   key travel_id     as TravelID,
18   key booking_id    as BookingID,
19       booking_date  as BookingDate,
20       customer_id   as CustomerID,
21       carrier_id    as CarrierID,
22       connection_id as ConnectionID,
23       flight_date   as FlightDate,
24       @Semantics.amount.currencyCode: 'CurrencyCode'
25       flight_price  as FlightPrice,
26       currency_code as CurrencyCode,
27
28
29
30       /* associations */
31
32       _Travel,
33       _Carrier,
34       _Customer,
35       _Connection,
36       _Flight
37
38 }
39
```

Vous **devez** maintenant sélectionner **les deux vues CDS** dans l'explorateur de projet et sélectionner **Activer** .

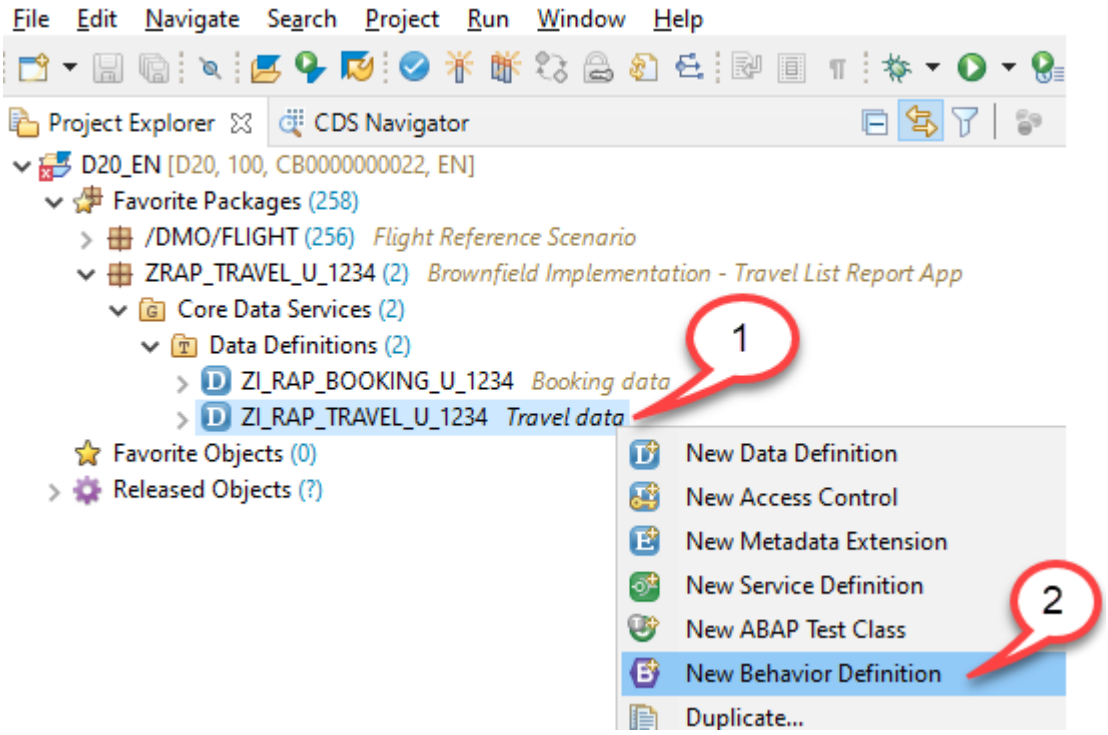
5.2 Définition et mise en œuvre du comportement des objets métier (Unité 2)

Objectif de l'unité

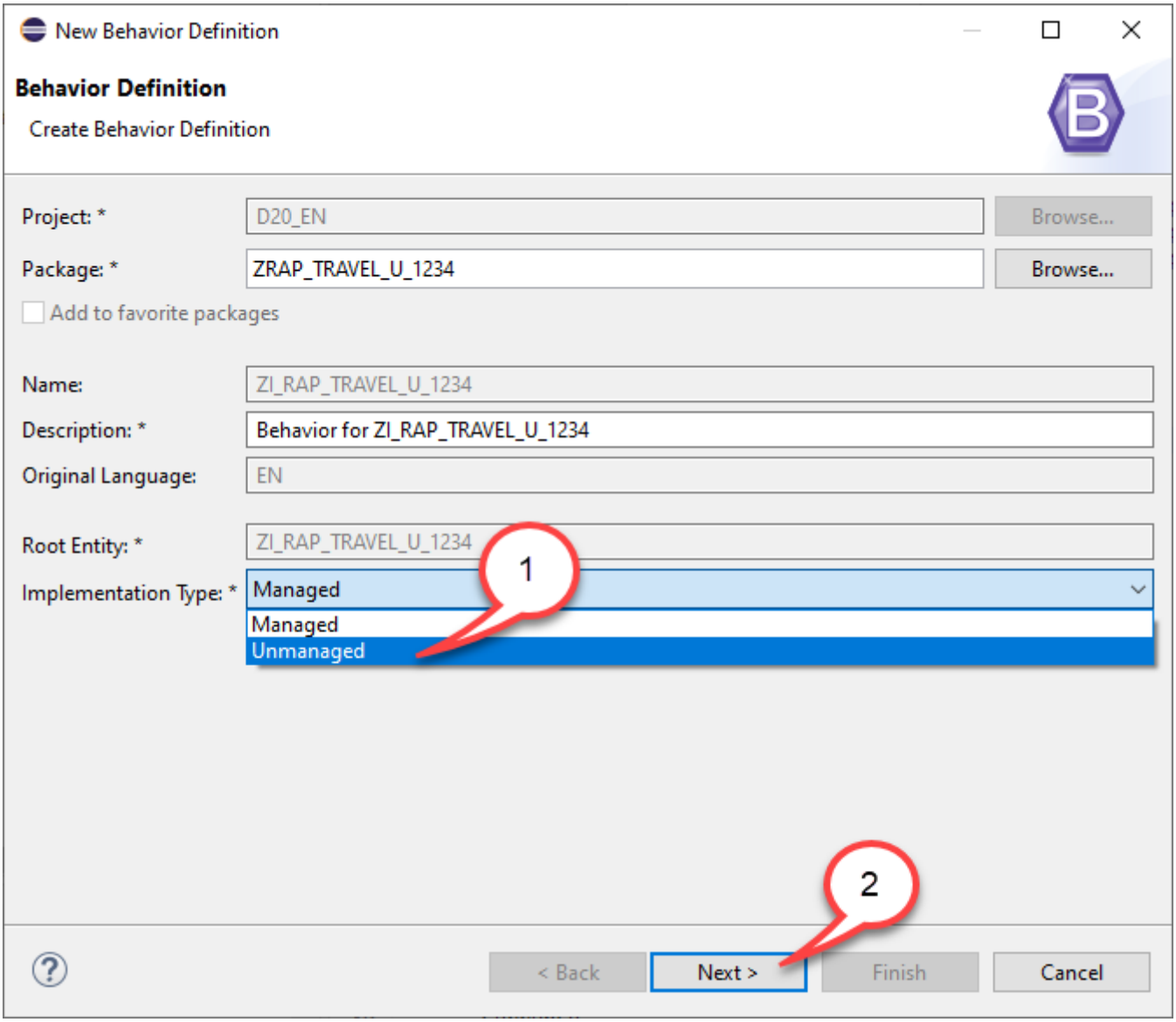
Définir et implémenter le **comportement** de l'objet métier ZI_RAP_Travel_U_#### dans un scénario **unmanaged**, en créant la **Behavior Definition (BDEF)** et la **classe d'implémentation (BIL)**, puis tester la méthode create via un **test unitaire avec EML**.

Étape 1 : Créer une définition de comportement pour la vue d'interface ZI_RAP_TRAVEL_####

1. Faites un clic droit sur la vue racine **ZI_RAP_Travel_U_####** de votre objet métier et choisissez **New Behavior Definition** dans le menu contextuel.



2. Ici, nous devons modifier le paramètre par défaut **Managed** du **type d'implémentation Unmanaged .** Le nom de la définition de comportement ne peut pas être modifié. Il doit être identique à celui de la vue racine de votre objet métier. Laissez les valeurs suggérées pour Package **ZRAP_TRAVEL_U_####** et Description **Behavior for ZI_RAP_TRAVEL_U_1234**inchangées, puis cliquez sur **Suivant** .



3. Sélectionnez une demande de transport ou créez-en une nouvelle.

4. L'assistant génère un fichier BDEF avec des paramètres par défaut que nous devons ajuster.

```
[D20] ZRAP_TRAVEL...   [D20] ZI_RAP_TRAV...   [D20] ZI_RAP_BOO...   [D
1  unmanaged implementation in class zbp_i_rap_travel_u_1234 unique;
2
3  define behavior for ZI_RAP_Travel_U_1234 //alias <alias_name>
4  //late numbering
5  //lock master
6  //etag master <field_name>
7  {
8    create;
9    update;
10   delete;
11   association _Booking { create; }
12 }
13
14 define behavior for ZI_RAP_Booking_U_1234 //alias <alias_name>
15 //late numbering
16 //lock dependent by <association>
17 //etag master <field_name>
18 {
19   create;
20   update;
21   delete;
22 }
```

5. Nous commençons donc par éditer la définition du comportement de l'entité Voyage.

- Au lieu d'utiliser une classe d'implémentation de comportement unique pour l'ensemble du BDEF, nous utiliserons des classes d'implémentation de comportement distinctes pour chaque entité de notre objet métier. Nous déplaçons donc l'instruction **implementation in class zbp_i_rap_travel_u_#### unique** juste après **define behavior for** celle de notre entité Travel.

```
[D20] ZRAP_TRAVEL...   [D20] ZI_RAP_TRAV...   [D20] ZI_RAP_BOO...   [D20
1  unmanaged implementation in class zbp_i_rap_travel_u_1234 unique;
2
3  define behavior for ZI_RAP_Travel_U_1234 //alias <alias_name>
4  //late numbering
5  //lock master
6  //etag master <field_name>
7  {
8    create;
```

```
1  unmanaged;
2
3  define behavior for ZI_RAP_Travel_U_1234 alias Travel
4  implementation in class zbp_i_rap_travel_u_1234 unique
```

- Nous ajoutons un alias **Travel**d ans notre définition de comportement pour la vue d'interface **ZI_RAP_Travel_U_####**

```
define behavior for ZI_RAP_Travel_U_9999 alias Travel
```

- Nous supprimons toute la ligne de code qui contient l'instruction suivante

  //late numbering

  puisque nous ne voulons pas utiliser de numérotation tardive

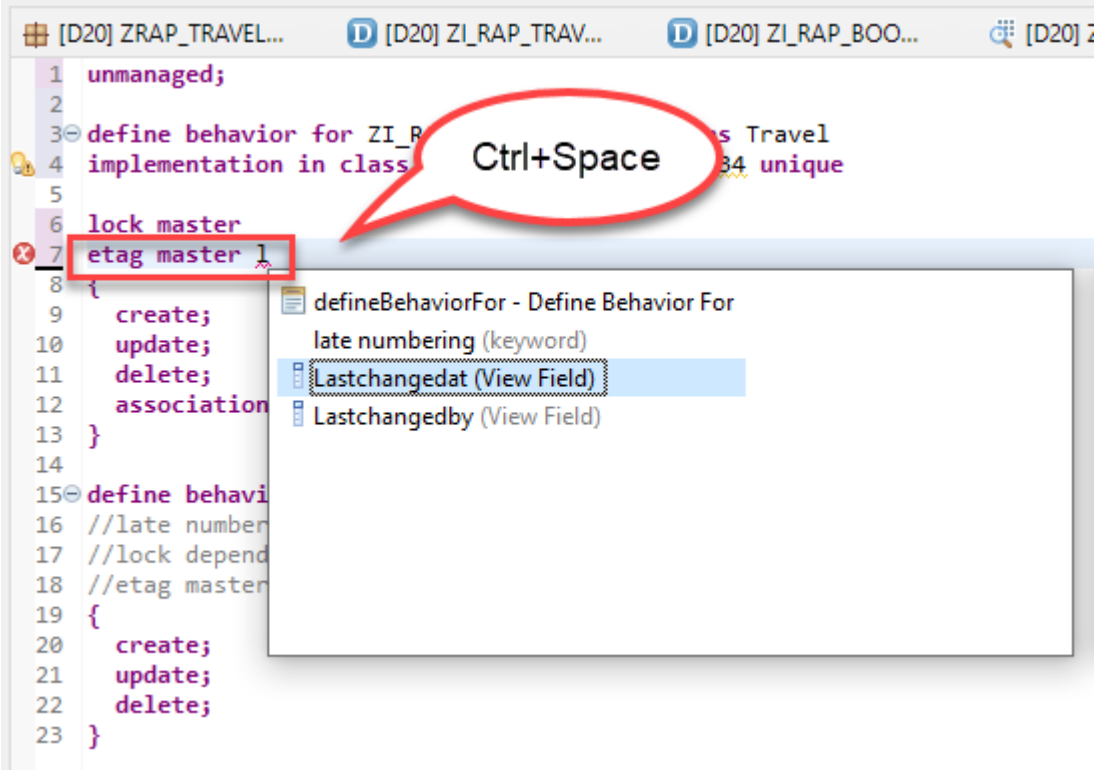
- Nous décommentons la déclaration

  Lock master

- Et nous décommentons la déclaration

  Etag master

En utilisant la saisie semi-automatique du code après la saisie **L**, nous pouvons sélectionner le champ **LastChangeAt**.



- Nous laissons les paramètres pour les opérations standard **Create**, **Update** et **Delete** nous laissons également le paramètre selon lequel l'association **_Booking** a été marquée comme créée activée.

6. Nous commençons maintenant à éditer le code dans notre définition de comportement pour l'entité Réservation **ZI_RAP_Booking_U_####**

- Ici, nous ajoutons également un alias **Booking** et nous ajoutons la déclaration **implementation in class zbp_i_rap_booking_u_#### unique** juste après la **define behavior for** déclaration pour notre entité de réservation.

- Nous supprimons à nouveau la déclaration de numérotation tardive

- Nous activons l'instruction **lock dependent by** et sélectionnons l'association **_Travel** via la complétion de code

```
14
15 define behavior for ZI_RAP_Booking_U_1234 alias Booking
16
17 lock dependent by
18 //etag master <fiel
19 {
20   create;
21   update;
22   delete;
23 }
```

defineBehaviorFor - Define Behavior For
_Carrier
_Connection
_Customer
_Flight
_Travel
1

- Étant donné que nous n'utilisons pas d'instruction principale etag mais choisissons l'option selon laquelle l'etag dépend de l'entité racine, nous devons définir**etag dependent by _Travel**

- Nous devons donc maintenant ajouter l'instruction **association _Travel** entre accolades

```
14
15 define behavior for ZI_RAP_Booking_U_1234 alias Booking
16
17 lock dependent by _Travel
18 etag dependent by _Travel
19 {
20   create;
21   update;
22   delete;
23
24   association _Travel;
25 }
```

11. Nous supprimons le comportement **create** de l'entité de réservation puisque les données de réservation ne doivent être créées qu'à partir **create by association** d'une entité de voyage parent existante.

12. Nous ajoutons maintenant un comportement pour ces champs, qui devraient devenir en lecture seule ou obligatoires

- Dans le code de définition du comportement pour l'entité de voyage, nous ajoutons (entre accolades)

```
  field ( read only ) TravelID;

    field ( mandatory ) AgencyID, CustomerID, BeginDate, EndDate;
```

- Dans le code de définition de comportement pour l'entité Réservation, nous ajoutons (entre accolades)

```
  field ( read only ) TravelID, BookingID;

    field ( mandatory ) BookingDate, CustomerID, CarrierId, ConnectionID,
FlightDate;
```

Le code de notre définition de comportement devrait maintenant ressembler à ce qui suit

```
unmanaged;



define behavior for ZI_RAP_Travel_U_#### alias Travel

implementation in class zbp_i_rap_travel_u_#### unique

lock master

etag master Lastchangedat

{

  create;

  update;

  delete;

  association _Booking { create; }



  field ( read only ) TravelID;

  field ( mandatory ) AgencyID, CustomerID, BeginDate, EndDate;
```

```
}


define behavior for ZI_RAP_Booking_U_#### alias Booking

implementation in class zbp_i_rap_booking_u_#### unique

lock dependent by _Travel

etag dependent by _Travel

{

  update;

  delete;

  association _Travel;


  field ( read only ) TravelID, BookingID;

  field ( mandatory ) BookingDate, CustomerID, CarrierId, ConnectionID,
FlightDate;


}
```

Étape 2 : Créer des structures de contrôle et des mappages

Créer des mappages

En guise d'étape finale, nous pouvons spécifier le mappage entre les noms de champs de la vue CDS et les noms de champs utilisés par notre logique métier héritée, c'est-à-dire par nos modules de fonction que nous utilisons pour créer, mettre à jour et supprimer les données de voyage.

1. Pour cela, nous devons ajouter le code suivant pour le mappage dans la définition de comportement de l'entité Travel.

```
1  unmanaged;
2
3  define behavior for ZI_RAP_Travel_U_1234 alias Travel
4  implementation in class zbp_i_rap_travel_u_1234 unique
5  lock master
6  etag master Lastchangedat
7  {
8    create;
9    update;
10   delete;
11   association _Booking { create; }
12
13   field ( read only ) TravelID;
14   field ( mandatory ) AgencyID, CustomerID, BeginDate, EndDate;
15
16
17   mapping for /DMO/TRAVEL control zsrap_travel_x_1234
18   {
19     TravelId = travel_id;
20     AgencyId = AGENCY_ID;
21     CustomerId = CUSTOMER_ID;
22     BeginDate = BEGIN_DATE;
23     EndDate = END_DATE;
24     BookingFee = BOOKING_FEE;
25     TotalPrice = TOTAL_PRICE;
26     CurrencyCode = CURRENCY_CODE;
27     Description = DESCRIPTION;
28     Status = STATUS;
29     Createdby = CREATEDBY;
30     Createdat = CREATEDAT;
31     Lastchangedby = LASTCHANGEDBY;
32     Lastchangedat = LASTCHANGEDAT;
33   }
34
35
36 }
```

1

```
mapping for /DMO/TRAVEL control zsrap_travel_x_####
  {
    TravelId = travel_id;
    AgencyId = AGENCY_ID;
    CustomerId = CUSTOMER_ID;
    BeginDate = BEGIN_DATE;
    EndDate = END_DATE;
    BookingFee = BOOKING_FEE;
    TotalPrice = TOTAL_PRICE;
    CurrencyCode = CURRENCY_CODE;
    Description = DESCRIPTION;
    Status = STATUS;
    Createdby = CREATEDBY;
    Createdat = CREATEDAT;
    Lastchangedby = LASTCHANGEDBY;
    Lastchangedat = LASTCHANGEDAT;
  }
```

2. Et nous ajoutons le code suivant pour le mappage dans la définition de comportement de l'entité Réservation.

```
38 define behavior for ZI_RAP_Booking_U_1234 alias Booking
39 implementation in class zbp_i_rap_booking_u_1234 unique
40 lock dependent by _Travel
41 etag dependent by _Travel
42 {
43   update;
44   delete;
45   association _Travel;
46
47   field ( read only ) TravelID, BookingID;
48   field ( mandatory ) BookingDate, CustomerID, CarrierId, Connect
49
50   mapping for /DMO/BOOKING control zsrap_booking_x_1234
51   {
52     TravelId = TRAVEL_ID;
53     BookingId = BOOKING_ID;
54     BookingDate = BOOKING_DATE;
55     CustomerId = CUSTOMER_ID;
56     CarrierId = CARRIER_ID;
57     ConnectionId = CONNECTION_ID;
58     FlightDate = FLIGHT_DATE;
59     FlightPrice = FLIGHT_PRICE;
60     CurrencyCode = CURRENCY_CODE;
61   }
62
63 }
```

1

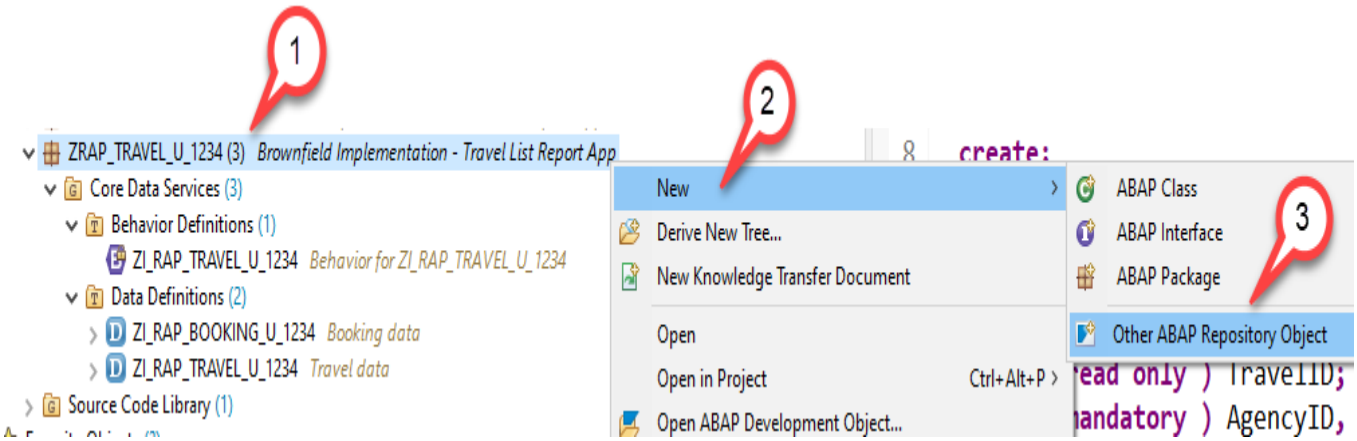
```
mapping for /DMO/BOOKING control zsrap_booking_x_####

  {

    TravelId = TRAVEL_ID;

    BookingId = BOOKING_ID;

    BookingDate = BOOKING_DATE;

    CustomerId = CUSTOMER_ID;

    CarrierId = CARRIER_ID;
```

```
    ConnectionId = CONNECTION_ID;

    FlightDate = FLIGHT_DATE;

    FlightPrice = FLIGHT_PRICE;

    CurrencyCode = CURRENCY_CODE;

  }
```

Créer des structures de contrôle

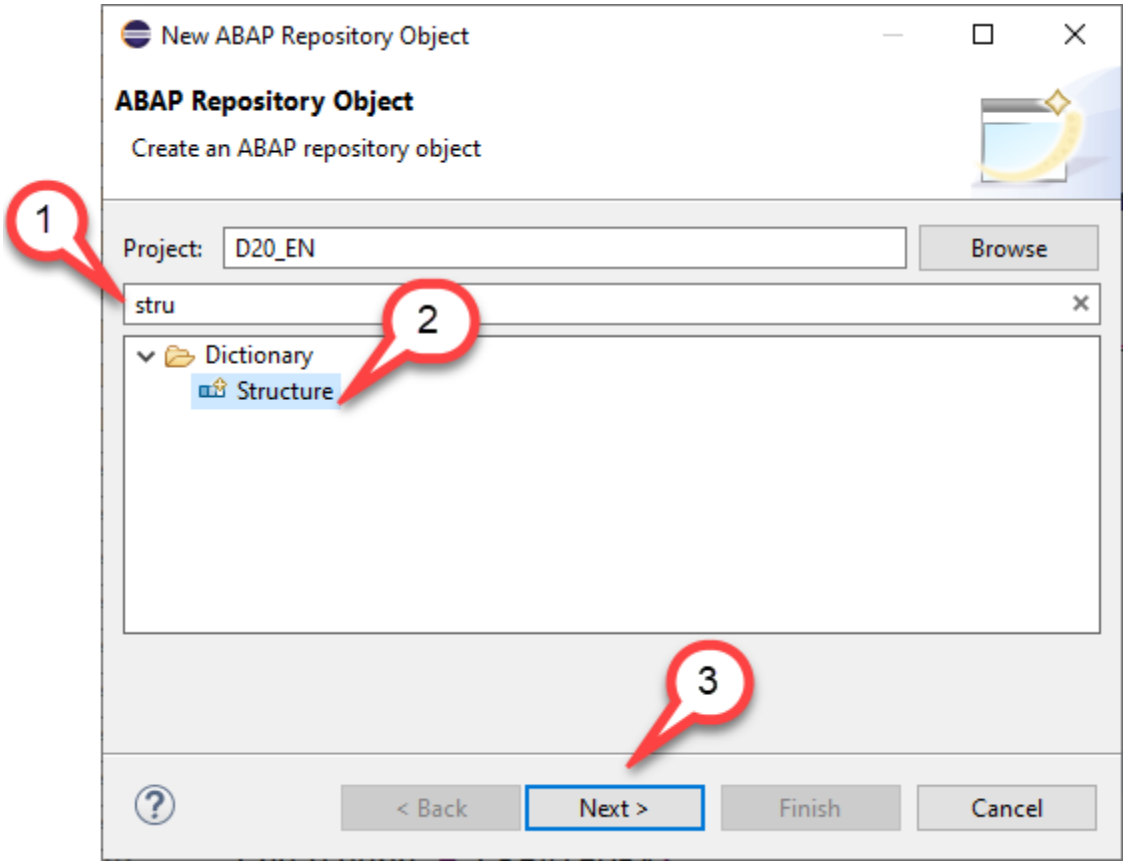
Il reste maintenant à créer deux structures de contrôle **zsrap_travel_x_####** nécessaires **zsrap_booking_x_####** aux mappages dans la définition du comportement.

1. Faites un clic droit sur le package et sélectionnez **Nouveau > Autre objet de référentiel ABAP**



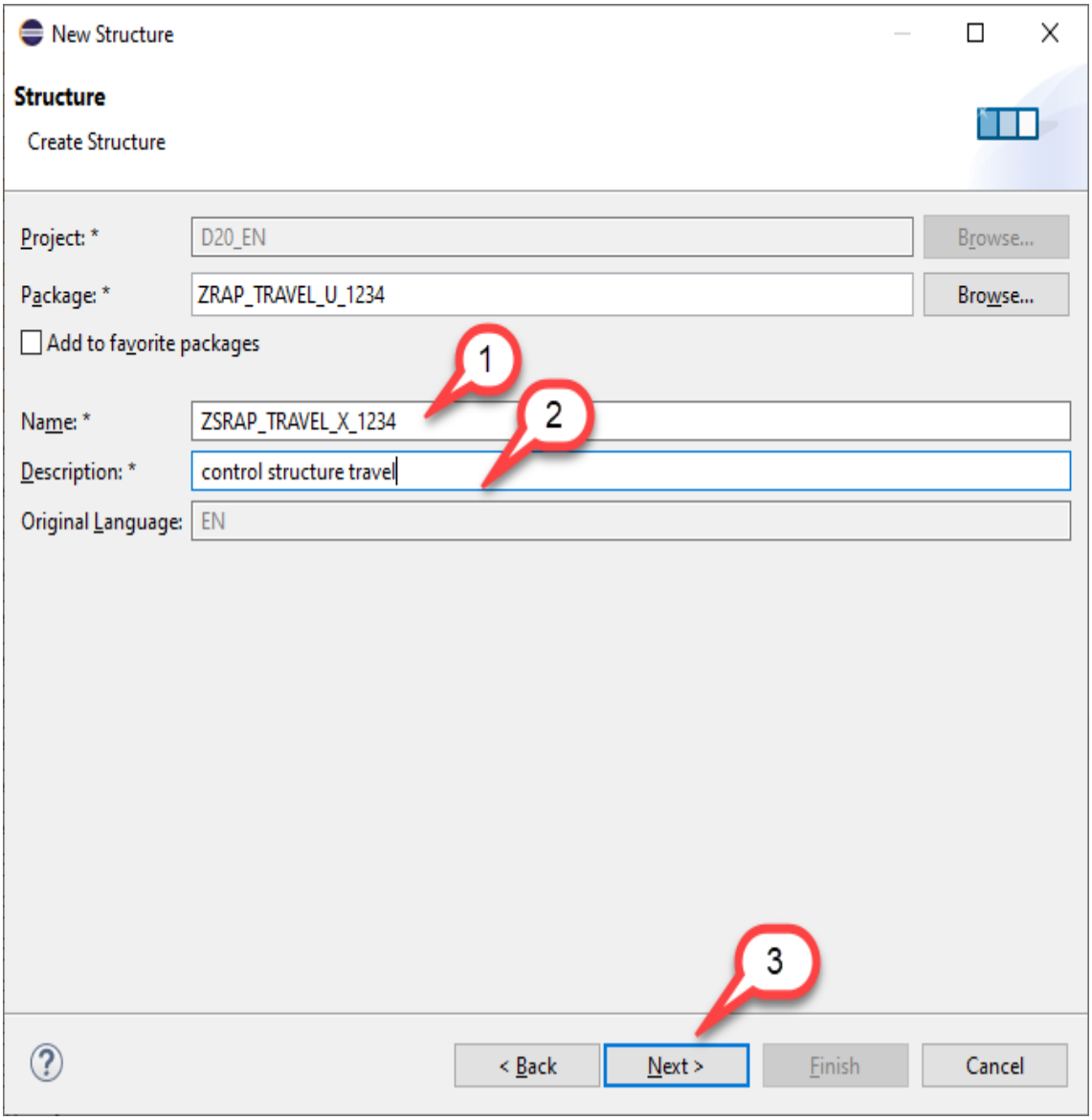
2. Dans l'assistant Nouvel objet de référentiel ABAP

   - Commencez à taperStructure
   - Dans la liste, sélectionnezStructure
   - Et appuyez sur **Suivant**



3. Entrez les valeurs suivantes

   - Nom:zsrap_travel_x_####
   - Description:Control structure travel
   - et appuyez sur **Suivant**



4. Sélection de la demande de transport

   - Sélectionnez une demande de transport
   - Appuyez sur **Terminer**

5. Dans l'éditeur de code source, remplacez le code du modèle par le code suivant

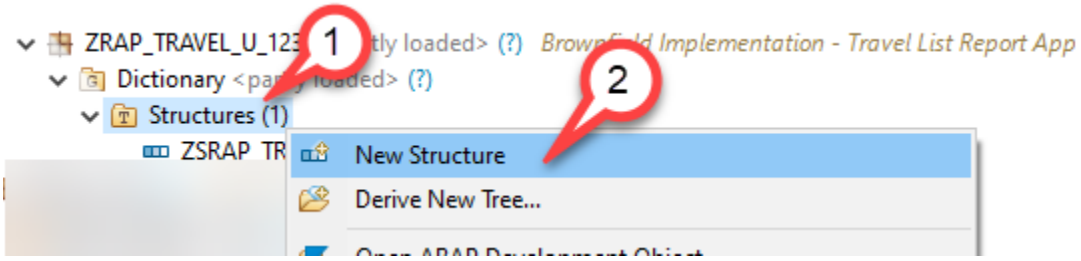
```
@EndUserText.label : 'Control structure for Travel data'

@AbapCatalog.enhancementCategory : #NOT_EXTENSIBLE
```

```
define structure zsrap_travel_x_#### {

  agency_id     : xsdboolean;

  customer_id   : xsdboolean;

  begin_date    : xsdboolean;

  end_date      : xsdboolean;

  booking_fee   : xsdboolean;

  total_price   : xsdboolean;

  currency_code : xsdboolean;

  description   : xsdboolean;

  status        : xsdboolean;


}
```

6. Enregistrez et activez vos modifications

7. Nous créons maintenant une deuxième structure de contrôle pour l'entité Réservation

   - Faites un clic droit sur le dossier **Structures**
   - Sélectionner **New Structure**



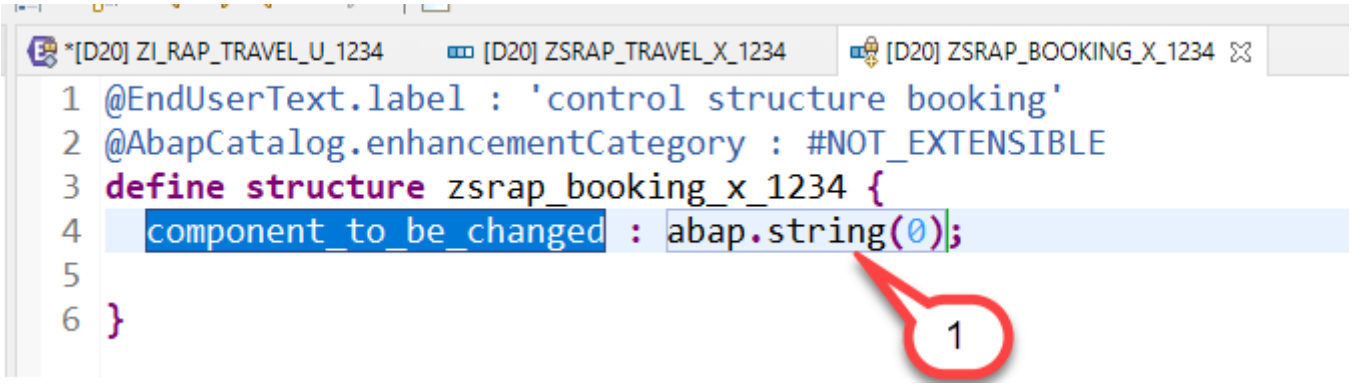
8. Entrez les valeurs suivantes

   - Nom : zsrap_booking_x_####
   - Description : control structure booking
   - Et appuyez sur **Suivant**

9. Sélection de la demande de transport

   - Sélectionnez une demande de transport et appuyez sur **Terminer**

10. Dans l'éditeur de code source, remplacez le code du modèle par le code suivant




```
@EndUserText.label : 'control structure for Booking'

@AbapCatalog.enhancementCategory : #NOT_EXTENSIBLE

define structure zsrap_booking_x_#### {

  booking_date   : xsdboolean;

  customer_id    : xsdboolean;

  carrier_id     : xsdboolean;

  connection_id : xsdboolean;

  flight_date    : xsdboolean;

  flight_price   : xsdboolean;

  currency_code : xsdboolean;


}
```

11. Enregistrez et activez vos modifications

```
unmanaged;

define behavior for ZI_RAP_Travel_U_1234 alias Travel

implementation in class zbp_i_rap_travel_u_1234 unique
```

```
lock master
etag master Lastchangedat
{
  create;
  update;
  delete;
  association _Booking { create; }

  field ( read only ) TravelID;
  field ( mandatory ) AgencyID, CustomerID, BeginDate, EndDate;

  mapping for /DMO/TRAVEL control zsrap_travel_x_1234
  {
    TravelId = travel_id;
    AgencyId = AGENCY_ID;
    CustomerId = CUSTOMER_ID;
    BeginDate = BEGIN_DATE;
    EndDate = END_DATE;
    BookingFee = BOOKING_FEE;
    TotalPrice = TOTAL_PRICE;
    CurrencyCode = CURRENCY_CODE;
    Description = DESCRIPTION;
    Status = STATUS;
    Createdby = CREATEDBY;
    Createdat = CREATEDAT;
    Lastchangedby = LASTCHANGEDBY;
```

```
    Lastchangedat = LASTCHANGEDAT;

  }


}


define behavior for ZI_RAP_Booking_U_1234 alias Booking

implementation in class zbp_i_rap_booking_u_1234 unique

lock dependent by _Travel

etag dependent by _Travel

{

  update;

  delete;

  association _Travel;


  field ( read only ) TravelID, BookingID;

  field ( mandatory ) BookingDate, CustomerID, CarrierId, ConnectionID,
FlightDate;


  mapping for /DMO/BOOKING control zsrap_booking_x_1234

  {

    TravelId = TRAVEL_ID;

    BookingId = BOOKING_ID;

    BookingDate = BOOKING_DATE;

    CustomerId = CUSTOMER_ID;

    CarrierId = CARRIER_ID;

    ConnectionId = CONNECTION_ID;

    FlightDate = FLIGHT_DATE;
```

```
    FlightPrice = FLIGHT_PRICE;

    CurrencyCode = CURRENCY_CODE;

  }



}
```

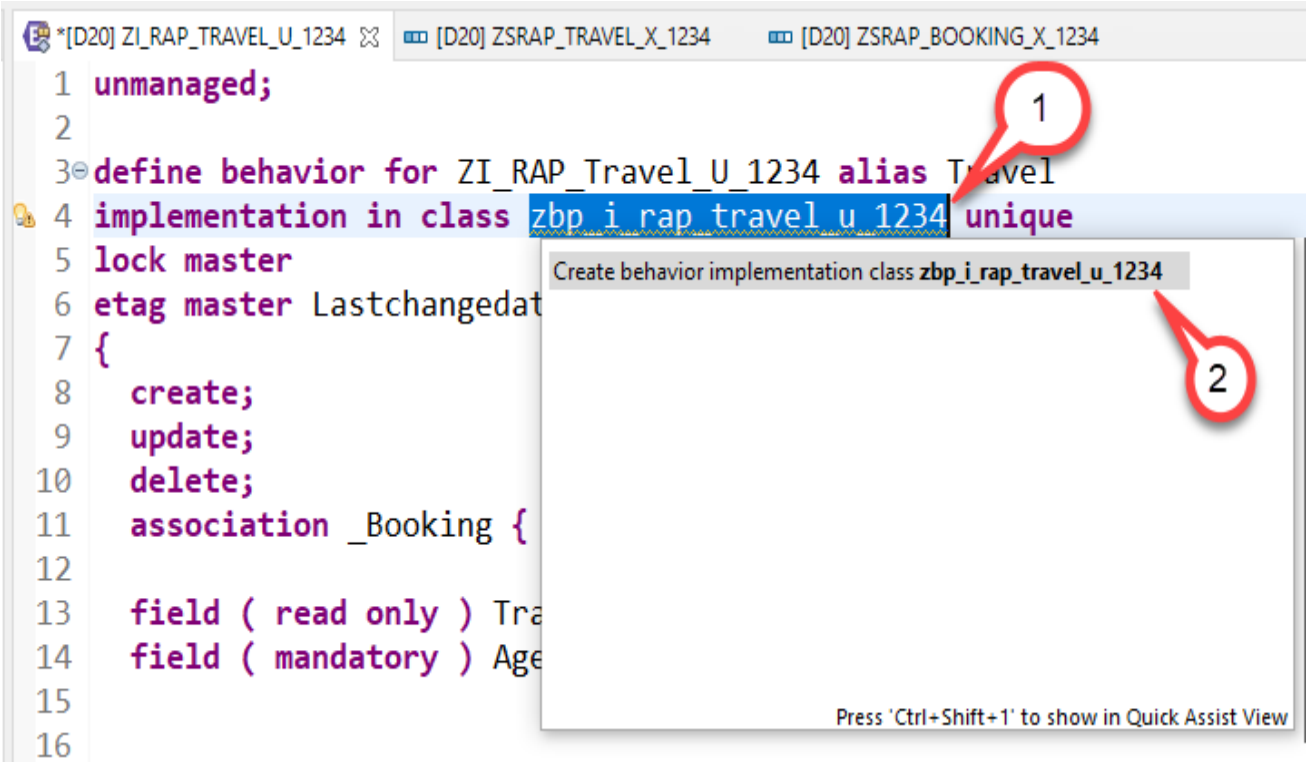
Étape 3 : Créer et implémenter la classe d'implémentation du comportement (BIL) pour l'entité Voyage

Nous créons maintenant la classe d'implémentation du comportement. Cependant, comme le code du BDEF n'est pas encore actif, nous devons d'abord l'activer. Ensuite, nous pouvons appuyer à nouveau sur Ctrl+1 pour créer la classe d'implémentation du comportement pour le déplacement.

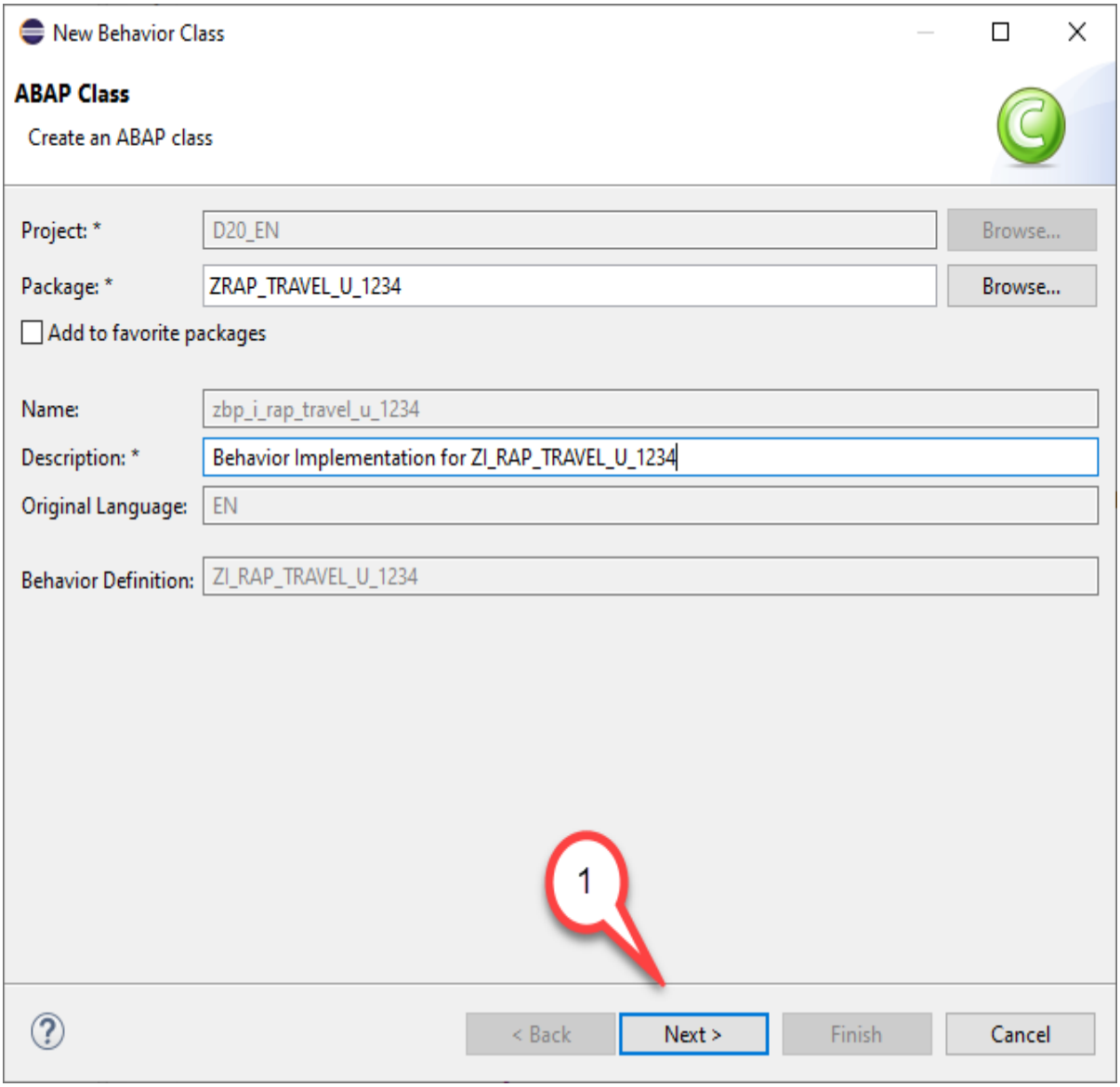
1. Activez votre définition de comportement en appuyant sur le bouton **Activer** si vous ne l'avez pas déjà fait.

2. Pour créer la classe d'implémentation du comportement de l'entité de voyage, nous pouvons utiliser une solution rapide : nous sélectionnons le nom de la classe dans la définition du comportement et appuyons sur Ctrl + 1.

    - Sélectionnez le nom de la classe d'implémentation du comportement zbp_i_rap_travel_u_####
    - Appuyez sur **Ctrl+1**
    - Double-cliquez sur la proposition Create behavior implementation class **zbp_i_rap_travel_u_####**



3. Créez la classe d'implémentation de comportement pour l'entité **Travel .**



4. Sélection d'une demande de transport
   - Sélectionnez une demande de transport
   - Appuyez sur **Terminer**
5. Vérifiez le modèle de code généré du Travel BIL :

   Par exemple vous voyez

   - Une méthode de création qui sera appelée lors de la création d'une entité de voyage et

- Une méthode de suppression si une entité de voyage doit être supprimée, à partir de la signature.

Veuillez noter qu'il s'agit d'une **classe locale** lhc_travel qui hérite de cl_abap_behavior_handler.

[D20] ZI_RAP_TRAVEL_U_1234 | [D20] ZSRAP_TRAVEL_X_1234 | [D20] ZSRAP_BOOKING_X_1234 | [D20] ZBP_I_RAP_TRAVEL_U_1234

ZBP_I_RAP_TRAVEL_U_1234 ▸ LHC_TRAVEL ▸

```
1  CLASS lhc_Travel DEFINITION INHERITING FROM cl_abap_behavior_handler.
2    PRIVATE SECTION.
3
4      METHODS create FOR MODIFY
5        IMPORTING entities FOR CREATE Travel.
6
7      METHODS delete FOR MODIFY
8        IMPORTING keys FOR DELETE Travel.
9
10     METHODS update FOR MODIFY
11       IMPORTING entities FOR UPDATE Travel.
12
13     METHODS lock FOR LOCK
14       IMPORTING keys FOR LOCK Travel.
15
16     METHODS read FOR READ
17       IMPORTING keys FOR READ Travel RESULT result.
18
19     METHODS cba_Booking FOR MODIFY
20       IMPORTING entities_cba FOR CREATE Travel\_Booking.
21
22     METHODS rba_Booking FOR READ
23       IMPORTING keys_rba FOR READ Travel\_Booking FULL result_requested RESULT resu
24
25 ENDCLASS.
26
27 CLASS lhc_Travel IMPLEMENTATION.
28
29   METHOD create.
```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

6. Nous commencerons par implémenter la méthode **create** et utiliserons le code préparé.

   - Accédez à la section d'implémentation de la classe

- Copiez et collez le code préparé pour la méthode de création dans le modèle de code source

```
ZBP_I_RAP_TRAVEL_U_1234 ▸ LHC_TRAVEL ▸ CREATE
25 ENDCLASS.
26
27 CLASS lhc_Travel IMPLEMENTATION.
28
29   METHOD create.
30
31
32     DATA messages   TYPE /dmo/t_message.
33     DATA legacy_entity_in  TYPE /dmo/travel.
34     DATA legacy_entity_out TYPE /dmo/travel.
35
36     LOOP AT entities ASSIGNING FIELD-SYMBOL(<entity>).
37
38       legacy_entity_in = CORRESPONDING #( <entity> MAPPING FROM ENTITY USING CONTROL ).
39
40       CALL FUNCTION '/DMO/FLIGHT_TRAVEL_CREATE'
41         EXPORTING
42           is_travel   = CORRESPONDING /dmo/s_travel_in( legacy_entity_in )
43         IMPORTING
44           es_travel   = legacy_entity_out
45           et_messages = messages.
46
47       IF messages IS INITIAL.
48         APPEND VALUE #( %cid = <entity>-%cid travelid = legacy_entity_out-travel_id ) TO mapped-travel.
49       ELSE.
50
51         "fill failed return structure for the framework
52         APPEND VALUE #( travelid = legacy_entity_in-travel_id ) TO failed-travel.
```

1

Le coda expliqué

Nous récupérons les données d'entrée de la charge utile via la table interne entities. Lors de la boucle sur ces données, nous utilisons le mappage central et la structure de contrôle pour spécifier les champs à utiliser pour créer les nouvelles entités de voyage. Les données sont déplacées vers la

structure legacy_entity_in structure, de même type que celle /DMO/FLIGHT_TRAVEL_CREATEacceptée par notre API existante. Lorsque la création de nouvelles données de voyage est réussie, l'API existante ne renvoie aucun message. Ce phénomène est typique des BAPI ; nous ajoutons alors l'identifiant de contenu **%cid** et le **travel_ID** à la structure **mapped-travel** . En cas de problème, les structures **failed** et **reported** sont renseignées afin que l'erreur puisse être affichée dans l'interface utilisateur.

```
 METHOD create.



    DATA messages   TYPE /dmo/t_message.

    DATA legacy_entity_in  TYPE /dmo/travel.

    DATA legacy_entity_out TYPE /dmo/travel.

    LOOP AT entities ASSIGNING FIELD-SYMBOL(<entity>).



      legacy_entity_in = CORRESPONDING #( <entity> MAPPING FROM ENTITY USING
CONTROL ).



      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_CREATE'

        EXPORTING

          is_travel   = CORRESPONDING /dmo/s_travel_in( legacy_entity_in )

        IMPORTING

          es_travel   = legacy_entity_out

          et_messages = messages.



      IF messages IS INITIAL.

        APPEND VALUE #( %cid = <entity>-%cid travelid = legacy_entity_out-
travel_id ) TO mapped-travel.

      ELSE.



        "fill failed return structure for the framework
```

```
        APPEND VALUE #( travelid = legacy_entity_in-travel_id ) TO failed-travel.

        "fill reported structure to be displayed on the UI

        APPEND VALUE #( travelid = legacy_entity_in-travel_id

                        %msg = new_message( id = messages[ 1 ]-msgid

                                            number = messages[ 1 ]-msgno

                                            v1 = messages[ 1 ]-msgv1

                                            v2 = messages[ 1 ]-msgv2

                                            v3 = messages[ 1 ]-msgv3

                                            v4 = messages[ 1 ]-msgv4

                                            severity = CONV #( messages[ 1 ]-msgty
) )

       ) TO reported-travel.




      ENDIF.


    ENDLOOP.


  ENDMETHOD.
```

7. Nous devons ensuite naviguer de la classe gestionnaire locale lhc_Travel, responsable de la maintenance des données dans le tampon, vers la classe sauvegarde locale lsc_ZI_RAP_Travel_U_####. Nous implémentons ici la méthode **save** , qui appelle l'API héritée, le module de fonction /DMO/FLIGHT_TRAVEL_SAVE, qui conserve les données du tampon partagé par les modules de fonction hérités, jusqu'à la base de données.

- Ajoutez le code suivant pour implémenter cette méthode **de sauvegarde**

```
  METHOD save.

    CALL FUNCTION '/DMO/FLIGHT_TRAVEL_SAVE'.

  ENDMETHOD.
```

```
102 CLASS lsc_ZI_RAP_Travel_U_1234 IMPLEMENTATION.
103
104   METHOD check_before_save.
105   ENDMETHOD.
106
107   METHOD finalize.
108   ENDMETHOD.
109
110   METHOD save.
111     CALL FUNCTION '/DMO/FLIGHT_TRAVEL_SAVE'.
112   ENDMETHOD.
113
114 ENDCLASS.
```

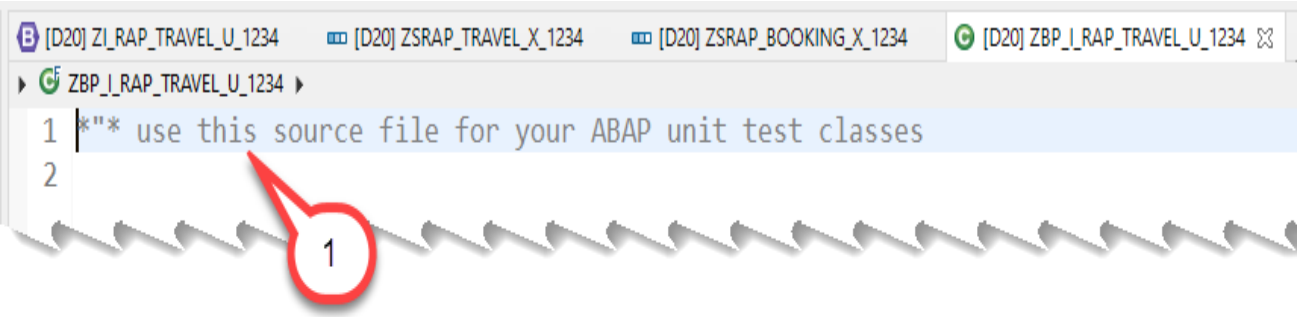
1

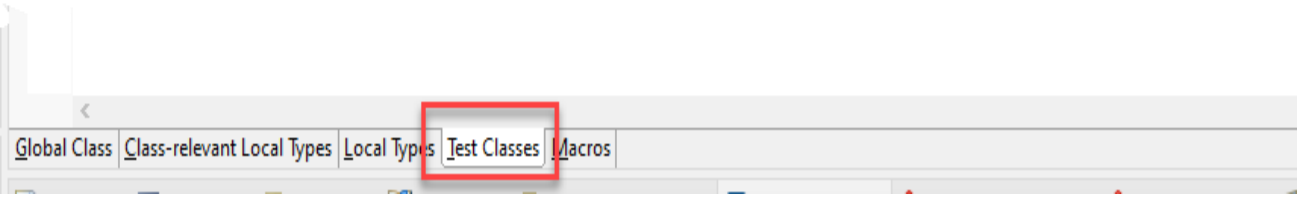
8. Enregistrez et activez vos modifications

Étape 4 : Test du BO avec EML

Comment allons-nous maintenant tester le fonctionnement de notre implémentation ? Nous pouvons procéder comme la dernière fois, c'est-à-dire publier le service, puis utiliser la fonctionnalité d'aperçu pour le tester. Nous allons ici présenter une méthode différente. Nous commencerons par implémenter un test unitaire, ce qui est toujours une bonne pratique.

Nous naviguons donc vers l'onglet test classes.





1. Implémenter la classe de test ltcl_integration_test

```
CLASS ltcl_integration_test DEFINITION FINAL FOR TESTING
   DURATION SHORT
   RISK LEVEL HARMLESS.

   PRIVATE SECTION.

      CLASS-DATA:
       cds_test_environment TYPE REF TO if_cds_test_environment.

      CLASS-METHODS:
       class_setup,
       class_teardown.
      METHODS:
       setup,
       teardown.
      METHODS:
       create_travel FOR TESTING RAISING cx_static_check.
  ENDCLASS.


  CLASS ltcl_integration_test IMPLEMENTATION.

   METHOD create_travel.

    DATA(today) = cl_abap_context_info=>get_system_date( ).
```

```
      DATA travels_in TYPE TABLE FOR CREATE zi_rap_travel_U_####\\travel.


      travels_in = VALUE #(     ( agencyid      = 070001   "Agency 070001 does
exist, Agency 1 does not exist

                                  customerid    = 1

                                  begindate     = today

                                  enddate       = today + 30

                                  bookingfee    = 30

                                  totalprice    = 330

                                  currencycode  = 'EUR'

                                  description   = |Test travel XYZ|

                                 ) ).


      MODIFY ENTITIES OF zi_rap_travel_u_####

        ENTITY travel

           CREATE FIELDS (    agencyid

                              customerid

                              begindate

                              enddate

                              bookingfee

                              totalprice

                              currencycode

                              description

                              Status )

             WITH travels_in

        MAPPED   DATA(mapped)

        FAILED   DATA(failed)
```

```
        REPORTED DATA(reported).


      cl_abap_unit_assert=>assert_initial( failed-travel ).

      cl_abap_unit_assert=>assert_initial( reported-travel ).

      COMMIT ENTITIES.


      DATA(new_travel_id) = mapped-travel[ 1 ]-TravelId.


      SELECT * FROM zi_rap_travel_u_#### WHERE TravelId = @new_travel_id INTO
TABLE @DATA(lt_travel)  .


      cl_abap_unit_assert=>assert_not_initial( lt_travel ).


      cl_abap_unit_assert=>assert_not_initial(

           VALUE #( lt_travel[  TravelID = new_travel_id ] OPTIONAL )

         ).

      cl_abap_unit_assert=>assert_equals(

          exp = 'N'

          act = lt_travel[ TravelID = new_travel_id ]-Status

        ).

    ENDMETHOD.


    METHOD class_setup.

      cds_test_environment = cl_cds_test_environment=>create_for_multiple_cds(

          i_for_entities = VALUE #( ( i_for_entity = 'zi_rap_travel_u_####' )

                                    ( i_for_entity = 'zi_rap_booking_u_####' ) )

                                 ).
```

```
    ENDMETHOD.


    METHOD class_teardown.
      cds_test_environment->destroy( ).
    ENDMETHOD.


    METHOD setup.
    ENDMETHOD.


    METHOD teardown.
      ROLLBACK ENTITIES.
      cds_test_environment->clear_doubles( ).
    ENDMETHOD.


ENDCLASS.
```

[D20] ZI_RAP_TRAVEL_U_1234 | [D20] ZSRAP_TRAVEL_X_1234 | [D20] ZSRAP_BOOKING_X_1234 | *[D20] ZBP_I_RAP_TRAVEL_U_1234

ZBP_I_RAP_TRAVEL_U_1234 ▸ LTCL_INTEGRATION_TEST ▸

```
1  CLASS ltcl_integration_test DEFINITION FINAL FOR TESTING
2    DURATION SHORT
3    RISK LEVEL HARMLESS.
4
5    PRIVATE SECTION.
6
7      CLASS-DATA:
8        cds_test_environment TYPE REF TO if_cds_test_environment.
9
10     CLASS-METHODS:
11       class_setup,
12       class_teardown.
13     METHODS:
14       setup,
15       teardown.
16     METHODS:
17       create_travel FOR TESTING RAISING cx_static_check.
18 ENDCLASS.
19
20
21 CLASS ltcl_integration_test IMPLEMENTATION.
22
23   METHOD create_travel.
24
25     DATA(today) = cl_abap_context_info=>get_system_date( ).
26     DATA travels_in TYPE TABLE FOR CREATE zi_rap_travel_U_####\\travel.
27
28     travels_in = VALUE #(     ( agencyid      = 070001   "Agency 070001 does exist, Agency 1 does not exist
29                                 customerid     = 1
```

1

[D20] ZI_RAP_TRAVEL_U_1234 | [D20] ZSRAP_TRAVEL_X_1234 | [D20] ZSRAP_BOOKING_X_1234 | *[D20] ZBP_

ZBP_I_RAP_TRAVEL_U_1234 ▸ LTCL_INTEGRATION_TEST ▸

```
1  CLASS ltcl_integration_test DEFINITION FINAL FOR TESTING
2    DURATION SHORT
3    RISK LEVEL HARMLESS.
4
5    PRIVATE SECTION.
6
7      CLASS-DATA:
8        cds_test_environment TYPE REF TO if_cds_test_environment.
9
10     CLASS-METHODS:
11       class_setup,
12       class_teardown.
13     METHODS:
14       setup,
15       teardown.
16     METHODS:
17       create_travel FOR TESTING RA
18 ENDCLASS.
19
20
21 CLASS ltcl_integration_test IMPLEM
22
23   METHOD create_travel.
24
25     DATA(today) = cl_abap_context_
26     DATA travels_in TYPE TABLE FOR CREATE zi_rap_travel_U_####\\tr
27
```

Find/Replace

1

Find: ####

Replace with: 1234

Direction: Forward / Backward

Scope: All / Selected lines

Options: Case sensitive / Wrap search / Whole word / Incremental / Regular expressions

Find | Replace/Find | Replace | Replace All | Close

2. Enregistrez et activez vos modifications

À quoi sert ce code ? Nous utilisons le framework de test CDS et créons des données pour un seul trajet. Pour cela, travels_in nous utilisons **EML (** *Entity Manipulation Language)* et l'instruction

```
MODIFY ENTITIES OF zi_rap_travel_u_####

      ENTITY travel
```

```
      CREATE FIELDS (    agencyid
                              customerid
                              begindate
                              enddate
                              bookingfee
                              totalprice
                              currencycode
                              description
                              Status )
             WITH travels_in
         MAPPED   DATA(mapped)
         FAILED   DATA(failed)
         REPORTED DATA(reported).
```

Utilisation des instructions assert

```
  cl_abap_unit_assert=>assert_initial( failed-travel ).
    cl_abap_unit_assert=>assert_initial( reported-travel ).
```

Nous vérifions que les structures de retour failedet reported, qui indiqueraient une erreur, sont vides. Si c'est le cas, nous validons les données dans les tables du framework de test double. Nous sélectionnons ensuite à nouveau les données dans l'interface CDS du framework de test double. Nous pouvons alors vérifier si les données ont été correctement créées.

Plus bas, nous voyons le code de la méthode class_setup, cette méthode crée le double framework de test pour les deux vues d'interface pour le voyage et la réservation.

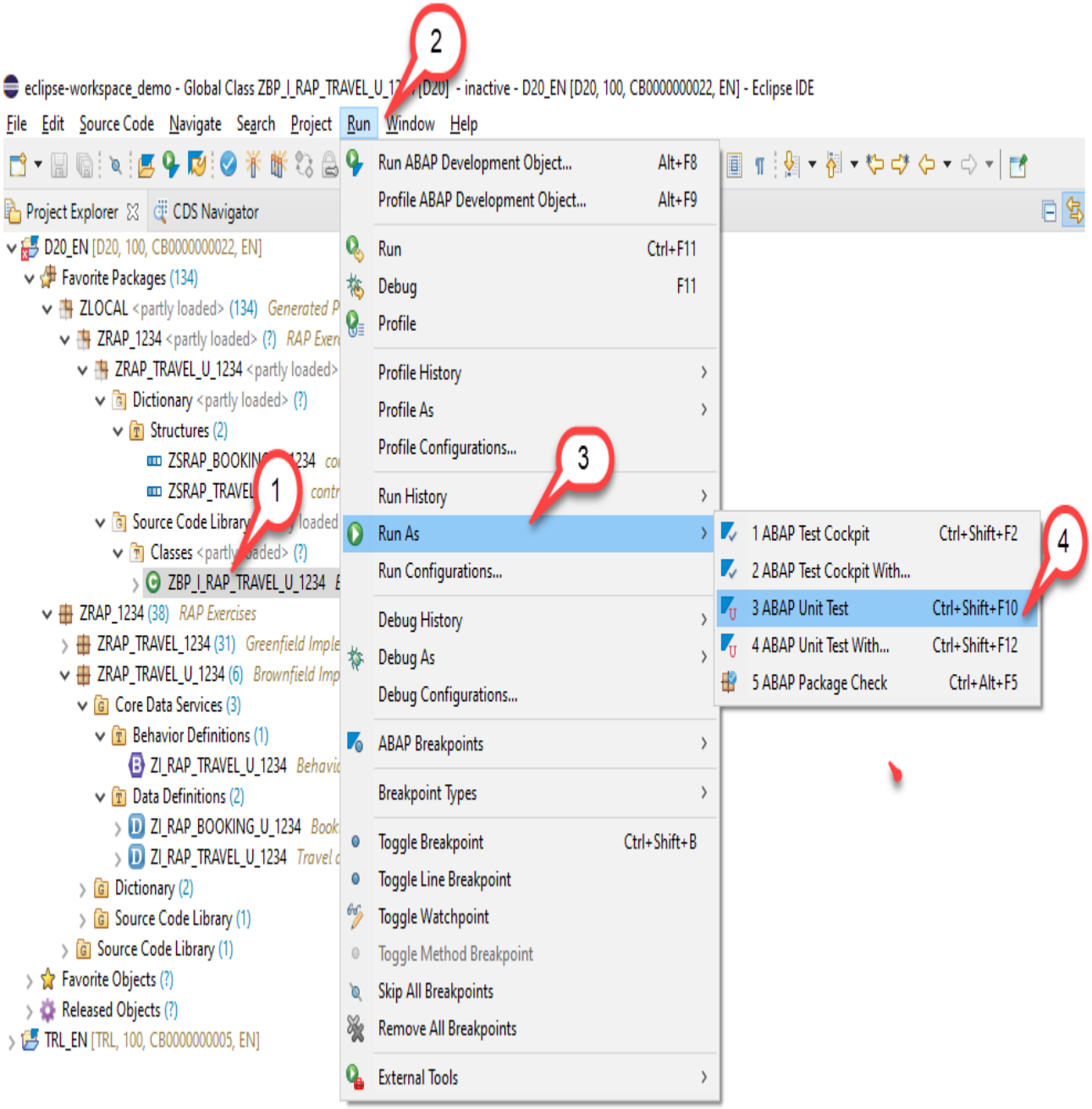
METHODE class_setup.

```
  METHOD class_setup.
    cds_test_environment = cl_cds_test_environment=>create_for_multiple_cds(
        i_for_entities = VALUE #( ( i_for_entity = 'zi_rap_travel_u_####' )
                                  ( i_for_entity = 'zi_rap_booking_u_####' ) )
                                 ).
```

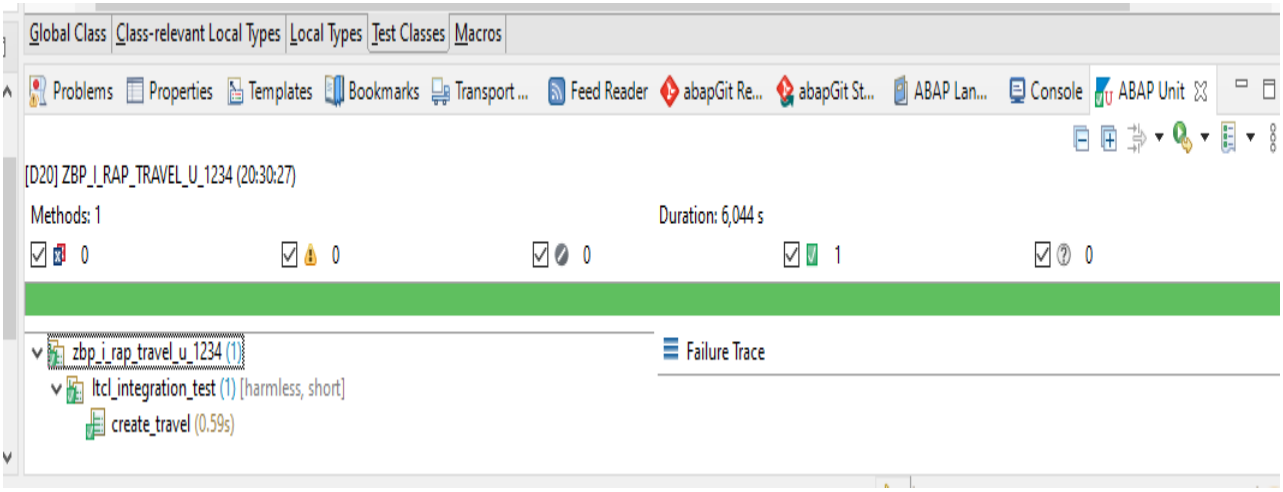
```
  ENDMETHOD.
```

Nous sélectionnons maintenant la classe dans l'explorateur de projet.

4. Dans le menu, choisissez **Exécuter > Exécuter en tant que > Test unitaire ABAP** ou utilisez le raccourci **CTRL+Maj+F10**



5. Vérifiez si le test a réussi.



Étape 5 : Implémenter les méthodes restantes pour l'entité Voyage

Il ne reste plus qu'à implémenter toutes les autres méthodes. Veuillez remplacer les méthodes de votre classe par un copier-coller des sections de code ci-dessous.

Supprimer

La méthode delete récupère simplement les clés des entités à supprimer. Dans la boucle, nous appelons le module de fonction /DMO/FLIGHT_TRAVEL_DELETE qui supprime une entité de déplacement. Une fois la suppression réussie, la valeur de la clé est ajoutée à la table interne mapped-travel.

```
METHOD delete.


    DATA messages TYPE /dmo/t_message.


    LOOP AT keys ASSIGNING FIELD-SYMBOL(<key>).


      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_DELETE'

        EXPORTING

          iv_travel_id = <key>-travelid

        IMPORTING

          et_messages  = messages.
```

```
      IF messages IS INITIAL.

        APPEND VALUE #( travelid = <key>-travelid ) TO mapped-travel.

      ELSE.

        "fill failed return structure for the framework
        APPEND VALUE #( travelid = <key>-travelid ) TO failed-travel.
        "fill reported structure to be displayed on the UI
        APPEND VALUE #( travelid = <key>-travelid
                        %msg = new_message( id = messages[ 1 ]-msgid
                                            number = messages[ 1 ]-msgno
                                            v1 = messages[ 1 ]-msgv1
                                            v2 = messages[ 1 ]-msgv2
                                            v3 = messages[ 1 ]-msgv3
                                            v4 = messages[ 1 ]-msgv4
                                            severity = CONV #( messages[ 1 ]-msgty
) )
       ) TO reported-travel.

      ENDIF.

    ENDLOOP.

  ENDMETHOD.
```

Enregistrez et activez vos modifications.

Mise à jour

Grâce à cette méthode update, nous pouvons mettre à jour les données. Nous obtenons ici à nouveau les données sous forme d'entités.
Les données sont à nouveau mappées selon notre mappage central et nous utilisons également la structure de contrôle.

Nous pouvons ensuite appeler le module de fonction de mise à jour de voyage /DMO/FLIGHT_TRAVEL_UPDATE.

```
METHOD update.



    DATA legacy_entity_in   TYPE /dmo/travel.

    DATA legacy_entity_x  TYPE /dmo/s_travel_inx . "refers to x structure (>
BAPIs)

    DATA messages TYPE /dmo/t_message.



    LOOP AT entities ASSIGNING FIELD-SYMBOL(<entity>).



      legacy_entity_in = CORRESPONDING #( <entity> MAPPING FROM ENTITY ).

      legacy_entity_x-travel_id = <entity>-TravelID.

      legacy_entity_x-_intx = CORRESPONDING zsrap_travel_x_####( <entity> MAPPING
FROM ENTITY ).



      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_UPDATE'

        EXPORTING

          is_travel   = CORRESPONDING /dmo/s_travel_in( legacy_entity_in )

          is_travelx  = legacy_entity_x

        IMPORTING

          et_messages = messages.
```

```
      IF messages IS INITIAL.

        APPEND VALUE #( travelid = legacy_entity_in-travel_id ) TO mapped-travel.

      ELSE.

        "fill failed return structure for the framework
        APPEND VALUE #( travelid = legacy_entity_in-travel_id ) TO failed-travel.
        "fill reported structure to be displayed on the UI
        APPEND VALUE #( travelid = legacy_entity_in-travel_id
                        %msg = new_message( id = messages[ 1 ]-msgid
                                            number = messages[ 1 ]-msgno
                                            v1 = messages[ 1 ]-msgv1
                                            v2 = messages[ 1 ]-msgv2
                                            v3 = messages[ 1 ]-msgv3
                                            v4 = messages[ 1 ]-msgv4
                                            severity = CONV #( messages[ 1 ]-msgty
) )
       ) TO reported-travel.

      ENDIF.


    ENDLOOP.

  ENDMETHOD.
```

Enregistrez et activez vos modifications

Verrouillage

Nous avons également implémenté la méthode lock, qui effectuerait un verrouillage afin d'éviter, lorsqu'une mise à jour est effectuée, que deux mises à jour se produisent en même temps.

```
 METHOD lock.



  "Instantiate lock object

    DATA(lock) = cl_abap_lock_object_factory=>get_instance( iv_name =
'/DMO/ETRAVEL' ).




    LOOP AT keys ASSIGNING FIELD-SYMBOL(<key>).

      TRY.

          "enqueue travel instance

          lock->enqueue(

              it_parameter  = VALUE #( (  name = 'TRAVEL_ID' value = REF #( <key>-
travelid ) ) )

          ).

          "if foreign lock exists

        CATCH cx_abap_foreign_lock INTO DATA(lx_foreign_lock).



          "fill failed return structure for the framework

          APPEND VALUE #( travelid = <key>-travelid ) TO failed-travel.

          "fill reported structure to be displayed on the UI

          APPEND VALUE #( travelid = <key>-travelid

                          %msg = new_message( id = '/DMO/CM_FLIGHT_LEGAC'

                                              number = '032'
```

```
                                                v1 = <key>-travelid

                                                v2 = lx_foreign_lock->user_name

                                                severity = CONV #( 'E' ) )

        ) TO reported-travel.


      ENDTRY.

    ENDLOOP.




  ENDMETHOD.
```

Enregistrez et activez vos modifications

Lire

La méthode read permet de lire les données depuis le tampon plutôt que depuis la base de données. Ceci est nécessaire, par exemple, lors des mises à jour, car nous utilisons des ETags. Le module fonction /DMO/FLIGHT_TRAVEL_READ permet de lire les données depuis le tampon de notre API héritée.

```
METHOD read.



    DATA: legacy_entity_out TYPE /dmo/travel,

          messages          TYPE /dmo/t_message.


    LOOP AT keys INTO DATA(key) GROUP BY key-TravelId.


      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_READ'

        EXPORTING

          iv_travel_id = key-travelid

        IMPORTING
```

```
          es_travel    = legacy_entity_out

          et_messages  = messages.


      IF messages IS INITIAL.

        "fill result parameter with flagged fields


        INSERT CORRESPONDING #( legacy_entity_out MAPPING TO ENTITY ) INTO TABLE
result.


      ELSE.


        "fill failed return structure for the framework

        APPEND VALUE #( travelid = key-travelid ) TO failed-travel.


        LOOP AT messages INTO DATA(message).


          "fill reported structure to be displayed on the UI

          APPEND VALUE #( travelid = key-travelid

                          %msg = new_message( id = message-msgid

                                              number = message-msgno

                                              v1 = message-msgv1

                                              v2 = message-msgv2

                                              v3 = message-msgv3

                                              v4 = message-msgv4

                                              severity = CONV #( message-msgty ) )
```

```
        ) TO reported-travel.

       ENDLOOP.

     ENDIF.



   ENDLOOP.




 ENDMETHOD.
```

Enregistrez et activez vos modifications

Créer par association - cba_Booking

Dans notre application de démonstration, nous supposons que les nouvelles réservations ne peuvent pas être créées séparément, mais uniquement conjointement avec une instance de voyage donnée. Le fait que les nouvelles réservations ne puissent être créées que pour une instance de voyage spécifique est pris en compte dans la définition du comportement par l'association _Booking :

```
association _Booking { create; }
```

Ce comportement est implémenté dans la méthode de réservation par association. Cette méthode est exécutée lors de la création d'une réservation via l'entité parente « Voyage ».

```
METHOD cba_Booking.



   DATA messages        TYPE /dmo/t_message.

   DATA lt_booking_old     TYPE /dmo/t_booking.

   DATA entity         TYPE /dmo/booking.

   DATA last_booking_id TYPE /dmo/booking_id VALUE '0'.


   LOOP AT entities_cba ASSIGNING FIELD-SYMBOL(<entity_cba>).


     DATA(travelid) = <entity_cba>-travelid.
```

```
CALL FUNCTION '/DMO/FLIGHT_TRAVEL_READ'
  EXPORTING
    iv_travel_id = travelid
  IMPORTING
    et_booking   = lt_booking_old
    et_messages  = messages.

IF messages IS INITIAL.

  IF lt_booking_old IS NOT INITIAL.

    last_booking_id = lt_booking_old[ lines( lt_booking_old ) ]-booking_id.

  ENDIF.

  LOOP AT <entity_cba>-%target ASSIGNING FIELD-SYMBOL(<entity>).

    entity = CORRESPONDING #( <entity> MAPPING FROM ENTITY USING CONTROL ) .

    last_booking_id += 1.
    entity-booking_id = last_booking_id.

    CALL FUNCTION '/DMO/FLIGHT_TRAVEL_UPDATE'
      EXPORTING
        is_travel   = VALUE /dmo/s_travel_in( travel_id = travelid )
```

```
              is_travelx  = VALUE /dmo/s_travel_inx( travel_id = travelid )

              it_booking  = VALUE /dmo/t_booking_in( ( CORRESPONDING #( entity ) ) )

              it_bookingx = VALUE /dmo/t_booking_inx(

                (

                  booking_id  = entity-booking_id

                  action_code = /dmo/if_flight_legacy=>action_code-create

                )

              )

            IMPORTING

              et_messages = messages.


          IF messages IS INITIAL.


            INSERT

              VALUE #(

                %cid = <entity>-%cid

                travelid = travelid

                bookingid = entity-booking_id

              )

              INTO TABLE mapped-booking.


          ELSE.




            INSERT VALUE #( %cid = <entity>-%cid travelid = travelid ) INTO TABLE failed-booking.
```

```
        LOOP AT messages INTO DATA(message) WHERE msgty = 'E' OR msgty = 'A'.


         INSERT

            VALUE #(

              %cid     = <entity>-%cid

              travelid = <entity>-TravelID

              %msg     = new_message(

                id       = message-msgid

                number   = message-msgno

                severity = if_abap_behv_message=>severity-error

                v1       = message-msgv1

                v2       = message-msgv2

                v3       = message-msgv3

                v4       = message-msgv4

              )

            )

            INTO TABLE reported-booking.


        ENDLOOP.


      ENDIF.


    ENDLOOP.


  ELSE.
```

```
        "fill failed return structure for the framework
        APPEND VALUE #( travelid = travelid ) TO failed-travel.
        "fill reported structure to be displayed on the UI
        APPEND VALUE #( travelid = travelid
                        %msg = new_message( id = messages[ 1 ]-msgid
                                            number = messages[ 1 ]-msgno
                                            v1 = messages[ 1 ]-msgv1
                                            v2 = messages[ 1 ]-msgv2
                                            v3 = messages[ 1 ]-msgv3
                                            v4 = messages[ 1 ]-msgv4
                                            severity = CONV #( messages[ 1 ]-msgty
) )
       ) TO reported-travel.




      ENDIF.

    ENDLOOP.



  ENDMETHOD.
```

Enregistrez et activez vos modifications

Lire par association - rba_Booking

Enfin, il existe un code pour la lecture par association des réservations, similaire à la méthode de lecture de l'entité Voyage. Cette méthode lit les données de réservation depuis le tampon.

```
METHOD rba_Booking.


    DATA: legacy_parent_entity_out  TYPE /dmo/travel,

          legacy_entities_out TYPE /dmo/t_booking,

          entity     LIKE LINE OF result,

          message     TYPE /dmo/t_message.




    LOOP AT keys_rba  ASSIGNING FIELD-SYMBOL(<key_rba>) GROUP  BY <key_rba>-
TravelId.


      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_READ'

        EXPORTING

          iv_travel_id = <key_rba>-travelid

        IMPORTING

          es_travel    = legacy_parent_entity_out

          et_booking   = legacy_entities_out

          et_messages  = message.


      IF message IS INITIAL.


        LOOP AT legacy_entities_out ASSIGNING FIELD-SYMBOL(<fs_booking>).

          "fill link table with key fields


          INSERT
```

```
          VALUE #(
              source-%key = <key_rba>-%key
              target-%key = VALUE #(
                TravelID  = <fs_booking>-travel_id
                BookingID = <fs_booking>-booking_id
            )
          )
          INTO TABLE  association_links .

      "fill result parameter with flagged fields
      IF result_requested = abap_true.

        entity = CORRESPONDING #( <fs_booking> MAPPING TO ENTITY ).
        INSERT entity INTO TABLE result.

      ENDIF.

    ENDLOOP.

  ELSE.
    "fill failed table in case of error

    failed-travel = VALUE #(
      BASE failed-travel
      FOR msg IN message (
        %key = <key_rba>-TravelID
```

```
          %fail-cause = COND #(
            WHEN msg-msgty = 'E' AND  ( msg-msgno = '016' OR msg-msgno = '009' )
            THEN if_abap_behv=>cause-not_found
            ELSE if_abap_behv=>cause-unspecific
          )
        )
      ).

    ENDIF.

  ENDLOOP.

ENDMETHOD.
```

Enregistrez et activez vos modifications

Étape 6 : Implémenter les méthodes de l'entité Réservation

Nous créons maintenant la classe d'implémentation du comportement pour l'entité Booking.

1. Activez votre définition de comportement en appuyant sur le bouton **Activer** si vous ne l'avez pas déjà fait.

2. Afin de créer la classe d'implémentation du comportement pour l'entité de réservation, nous pouvons également utiliser une solution rapide.

   - Sélectionnez le nom de la classe d'implémentation du comportement zbp_i_rap_booking_u_####
   - Appuyez sur **Ctrl+1**
   - Double-cliquez sur la proposition Create behavior implementation class **zbp_i_rap_booking_u_####**

```
[D20] ZI_RAP_TRAVEL_U_1234   [D20] /DMO/FLIGHT_TRAVEL_UPDATE
19      TravelId = travel_id;
20      AgencyId = AGENCY_ID;
21      CustomerId = CUSTOMER_ID;
22      BeginDate = BEGIN_DATE;
23      EndDate = END_DATE;
24      BookingFee = BOOKING_FEE;
25      TotalPrice = TOTAL_PRICE;
26      CurrencyCode = CURRENCY_CODE;
27      Description = DESCRIPTION;
28      Status = STATUS;
29      Createdby = CREATEDBY;
30      Createdat = CREATEDAT;
31      Lastchangedby = LASTCHANGEDBY;
32      Lastchangedat = LASTCHANGEDAT;
33    }
34
35
36  }
37
38  define behavior for ZI_RAP_Booking_U_1234 alias Booking
39  implementation in class zbp_i_rap_booking_u_1234 unique
    Create global behavior implmentation class zbp_i_rap_booking_u_1234 for
    behavior zi_rap_travel_u_1234.
    Press 'Tab' from proposal table or click for focus
    Create behavior implementation class zbp_i_rap_booking_u_1234
    Press 'Ctrl+Shift+1' to show in Quick Assist View
54      BookingDate = BOOKING_DATE;
55      CustomerId = CUSTOMER_ID;
56      CarrierId = CARRIER_ID;
57      ConnectionId = CONNECTION_ID;
58      FlightDate = FLIGHT_DATE;
59      FlightPrice = FLIGHT_PRICE;
60      CurrencyCode = CURRENCY_CODE;
61    }
62
63  }
```

1

2

3. Dans la boîte de dialogue **Nouvelle classe de comportement,** créez la classe d'implémentation de comportement pour l'entité **Réservation** et appuyez sur **Suivant** .

4. Sélection d'une demande de transport

   - Sélectionnez une demande de transport
   - Appuyez sur **Terminer**

5. Vérifiez le modèle de code généré du Bookin BIL :

   par exemple vous voyez

- une méthode de suppression qui sera appelée lors de la suppression d'une entité de réservation et une méthode de mise à jour si une entité de réservation doit être mise à jour, à partir de la signature.

Veuillez noter qu'il s'agit d'une **classe locale** lhc_Booking qui hérite de cl_abap_behavior_handler.

```
[D20] ZI_RAP_TRAVEL_U_1234   [D20] /DMO/FLIGHT_TRAVEL_UPDATE   [D20] ZBP_I_RAP_BOOKING_U_1234
ZBP_I_RAP_BOOKING_U_1234 ▸ LHC_BOOKING ▸
1  CLASS lhc_Booking DEFINITION INHERITING FROM cl_abap_behavior_handler.
2    PRIVATE SECTION.
3
4      METHODS delete FOR MODIFY
5        IMPORTING keys FOR DELETE Booking.
6
7      METHODS update FOR MODIFY
8        IMPORTING entities FOR UPDATE Booking.
9
10     METHODS read FOR READ
11       IMPORTING keys FOR READ Booking RESULT result.
12
13     METHODS rba_Travel FOR READ
14       IMPORTING keys_rba FOR READ Booking\_Travel FULL result_requested RESULT result LINK association_links.
15
16 ENDCLASS.
17
18 CLASS lhc_Booking IMPLEMENTATION.
19
20   METHOD delete.
21   ENDMETHOD.
22
23   METHOD update.
24   ENDMETHOD.
25
26   METHOD read.
27   ENDMETHOD.
28
29   METHOD rba_Travel.
30   ENDMETHOD.
31
32 ENDCLASS.
Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros
```

6. Nous commencerons par implémenter la méthode **delete** et utiliserons le code préparé.

Supprimer

Les méthodes de suppression obtiennent la liste des clés travelid et bookingid des entités de réservation à supprimer. Pour supprimer des entités de réservation à l'aide de l'API héritée, nous utiliserons le module de fonction /DMO/FLIGHT_TRAVEL_UPDATE.

```
METHOD delete.
```

```
    DATA messages TYPE /dmo/t_message.

    LOOP AT keys ASSIGNING FIELD-SYMBOL(<key>).

      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_UPDATE'
        EXPORTING
          is_travel   = VALUE /dmo/s_travel_in( travel_id = <key>-travelid )
          is_travelx  = VALUE /dmo/s_travel_inx( travel_id = <key>-travelid )
          it_booking  = VALUE /dmo/t_booking_in( ( booking_id = <key>-bookingid )
)
          it_bookingx = VALUE /dmo/t_booking_inx( ( booking_id  = <key>-bookingid
                                                    action_code =
/dmo/if_flight_legacy=>action_code-delete ) )
        IMPORTING
          et_messages = messages.

      IF messages IS INITIAL.

        APPEND VALUE #( travelid = <key>-travelid
                        bookingid = <key>-bookingid ) TO mapped-booking.

      ELSE.

        "fill failed return structure for the framework
        APPEND VALUE #( travelid = <key>-travelid
                         bookingid = <key>-bookingid ) TO failed-booking.
```

```
        LOOP AT messages INTO DATA(message).
          "fill reported structure to be displayed on the UI
          APPEND VALUE #( travelid = <key>-travelid
                          bookingid = <key>-bookingid
                  %msg = new_message( id = message-msgid
                                      number = message-msgno
                                      v1 = message-msgv1
                                      v2 = message-msgv2
                                      v3 = message-msgv3
                                      v4 = message-msgv4
                                      severity = CONV #( message-msgty )
)
         ) TO reported-booking.
        ENDLOOP.




      ENDIF.


    ENDLOOP.


  ENDMETHOD.
```

Enregistrez et activez vos modifications

Mise à jour

La méthode de mise à jour obtient une table de type FOR UPDATE Booking.

```
METHOD update.


    DATA messages TYPE /dmo/t_message.

    DATA legacy_entity_in  TYPE /dmo/booking.

    DATA legacy_entity_x TYPE /dmo/s_booking_inx.




    LOOP AT entities ASSIGNING FIELD-SYMBOL(<entity>).


      legacy_entity_in = CORRESPONDING #( <entity> MAPPING FROM ENTITY ).


      legacy_entity_x-booking_id = <entity>-BookingID.

      legacy_entity_x-_intx      = CORRESPONDING zsrap_booking_x_####( <entity>
MAPPING FROM ENTITY ).

      legacy_entity_x-action_code = /dmo/if_flight_legacy=>action_code-update.


      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_UPDATE'

        EXPORTING

          is_travel   = VALUE /dmo/s_travel_in( travel_id = <entity>-travelid )

          is_travelx  = VALUE /dmo/s_travel_inx( travel_id = <entity>-travelid )

          it_booking  = VALUE /dmo/t_booking_in( ( CORRESPONDING #(
legacy_entity_in ) ) )

          it_bookingx = VALUE /dmo/t_booking_inx( ( legacy_entity_x ) )

        IMPORTING

          et_messages = messages.
```

```
      IF messages IS INITIAL.


        APPEND VALUE #( travelid = <entity>-travelid

                        bookingid = legacy_entity_in-booking_id ) TO mapped-
booking.


      ELSE.


        "fill failed return structure for the framework

        APPEND VALUE #( travelid = <entity>-travelid

                         bookingid = legacy_entity_in-booking_id ) TO failed-
booking.

        "fill reported structure to be displayed on the UI


        LOOP AT messages INTO DATA(message).

          "fill reported structure to be displayed on the UI

          APPEND VALUE #( travelid = <entity>-travelid

                          bookingid = legacy_entity_in-booking_id

                  %msg = new_message( id = message-msgid

                                                  number = message-msgno

                                                  v1 = message-msgv1

                                                  v2 = message-msgv2

                                                  v3 = message-msgv3

                                                  v4 = message-msgv4

                                                  severity = CONV #( message-msgty )
)

         ) TO reported-booking.
```

```
        ENDLOOP.


      ENDIF.


    ENDLOOP.



  ENDMETHOD.
```

Enregistrez et activez vos modifications

Lire

La méthode read récupère la liste des clés des entités de réservation dont les données doivent être lues. L'API héritée, le module de fonction, /DMO/FLIGHT_TRAVEL_READ lit systématiquement toutes les réservations pour un TravelID. Afin d'éviter d'appeler l'API héritée, nous regroupons plusieurs fois les clés entrantes par TravelID.

```
METHOD read.


    DATA: legacy_parent_entity_out TYPE /dmo/travel,

          legacy_entities_out      TYPE /dmo/t_booking,

          messages                 TYPE /dmo/t_message.


    "Only one function call for each requested travelid

    LOOP AT keys ASSIGNING FIELD-SYMBOL(<key_parent>)

                            GROUP BY <key_parent>-travelid .


      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_READ'

        EXPORTING
```

```
          iv_travel_id = <key_parent>-travelid

        IMPORTING

          es_travel    = legacy_parent_entity_out

          et_booking   = legacy_entities_out

          et_messages  = messages.


      IF messages IS INITIAL.

        "For each travelID find the requested bookings

        LOOP AT GROUP <key_parent> ASSIGNING FIELD-SYMBOL(<key>)

                                       GROUP BY <key>-%key.


          READ TABLE legacy_entities_out INTO DATA(legacy_entity_out) WITH KEY
travel_id  = <key>-%key-TravelID

                                                                    booking_id =
<key>-%key-BookingID .

          "if read was successfull

          "fill result parameter with flagged fields

          IF sy-subrc = 0.


            INSERT CORRESPONDING #( legacy_entity_out MAPPING TO ENTITY ) INTO
TABLE result.


          ELSE.

            "BookingID not found

            INSERT

              VALUE #( travelid    = <key>-TravelID

                       bookingid   = <key>-BookingID

                       %fail-cause = if_abap_behv=>cause-not_found )
```

```
              INTO TABLE failed-booking.

          ENDIF.

        ENDLOOP.

      ELSE.

        "TravelID not found or other fail cause

        LOOP AT GROUP <key_parent> ASSIGNING <key>.

          failed-booking = VALUE #(  BASE failed-booking

                                     FOR msg IN messages ( %key-TravelID    =
<key>-TravelID

                                                           %key-BookingID   =
<key>-BookingID

                                                           %fail-cause      =
COND #( WHEN msg-msgty = 'E' AND ( msg-msgno = '016' OR msg-msgno = '009' )

THEN if_abap_behv=>cause-not_found

ELSE if_abap_behv=>cause-unspecific ) ) ).

        ENDLOOP.


      ENDIF.


    ENDLOOP.


  ENDMETHOD.
```

Enregistrez et activez vos modifications

rba_travel

Cette méthode lit les détails d'une entité de voyage parent lorsqu'une entité de réservation est mise à jour car nous avons défini dans la définition de comportement que l'etag de l'entité de réservation dépend de l'etag de l'entité parent Travel.

```
define behavior for ZI_RAP_Booking_U_1234 alias Booking

implementation in class zbp_i_rap_booking_u_1234 unique

lock dependent by _Travel

etag dependent by _Travel
```

Regardons la signature de la méthode rba_travel (lecture par association)

```
METHODS rba_Travel FOR READ

      IMPORTING

         keys_rba FOR READ Booking\_Travel FULL result_requested RESULT result
LINK association_links.
```

Nous voyons que le paramètre d'importation keys_rba est essentiellement une table qui contient les deux champs clés travelid et bookingid.
De plus, il existe un indicateur result_requested qui contrôle si seules les paires de clés de l'association doivent être renvoyées dans une table association_links ou si la cible de l'association avec tous les champs doit être renvoyée dans la table results.
L'API héritée /dmo/flight_travel_read permettant de lire les détails d'une entité Voyage et de ses réservations associées a été implémentée de telle sorte que les informations sur une entité Réservation ne peuvent être récupérées qu'en fournissant le TravelID de l'entité parent.

```
/dmo/flight_travel_read

  importing      iv_travel_id     type /dmo/travel_id

         iv_include_buffer        type abap_boolean default abap_true

  exporting      es_travel        type /dmo/travel

         et_booking       type /dmo/t_booking

         et_booking_supplement    type /dmo/t_booking_supplement

    et_messages  type /dmo/t_message
```

Nous devons donc d'abord regrouper les résultats par travelID afin d'éviter d'appeler plusieurs fois l'API héritée pour le même TravelID.

```
 LOOP AT keys_rba ASSIGNING FIELD-SYMBOL(<fs_travel>)

                                GROUP BY <fs_travel>-TravelID.
```

```
      CALL FUNCTION '/DMO/FLIGHT_TRAVEL_READ'
        EXPORTING
          iv_travel_id = <fs_travel>-%key-TravelID
        IMPORTING
          es_travel    = ls_travel_out
          et_messages  = lt_message.

        ….

  ENDLOOP.
```

La déclaration

```
LOOP AT GROUP <fs_travel> ASSIGNING FIELD-SYMBOL(<fs_booking>).
…
ENDLOOP.
```

puis fournit les entités de réservation pour un TravelID.

```
METHOD rba_Travel.

    DATA: ls_travel_out  TYPE /dmo/travel,
          lt_booking_out TYPE /dmo/t_booking,
          ls_travel      LIKE LINE OF result,
          lt_message     TYPE /dmo/t_message.

    "result  type table for read result /dmo/i_travel_u\\booking\_travel
```

```
"Only one function call for each requested travelid
LOOP AT keys_rba ASSIGNING FIELD-SYMBOL(<fs_travel>)
                             GROUP BY <fs_travel>-TravelID.


  CALL FUNCTION '/DMO/FLIGHT_TRAVEL_READ'
    EXPORTING
      iv_travel_id = <fs_travel>-%key-TravelID
    IMPORTING
      es_travel    = ls_travel_out
      et_messages  = lt_message.


  IF lt_message IS INITIAL.


    LOOP AT GROUP <fs_travel> ASSIGNING FIELD-SYMBOL(<fs_booking>).
      "fill link table with key fields
      INSERT VALUE #( source-%key = <fs_booking>-%key
                      target-%key = ls_travel_out-travel_id )
       INTO TABLE association_links .


      IF  result_requested  = abap_true.
        "fill result parameter with flagged fields
        ls_travel = CORRESPONDING #(  ls_travel_out MAPPING TO ENTITY ).
        INSERT ls_travel INTO TABLE result.
      ENDIF.
    ENDLOOP.
```

```
      ELSE. "fill failed table in case of error

        failed-booking = VALUE #(  BASE failed-booking

                                FOR msg IN lt_message ( %key-TravelID    =
<fs_travel>-%key-TravelID

                                                        %key-BookingID   =
<fs_travel>-%key-BookingID

                                                        %fail-cause      = COND #(
WHEN msg-msgty = 'E' AND ( msg-msgno = '016' OR msg-msgno = '009' )


THEN if_abap_behv=>cause-not_found


ELSE if_abap_behv=>cause-unspecific ) ) ).

      ENDIF.



    ENDLOOP.



  ENDMETHOD.
```

Enregistrez et activez vos modifications

Résumé

Dans cette unité, vous avez défini un objet métier RAP **non géré et vous avez implémenté des opérations, à savoir Créer**, **Mettre à jour**, **Supprimer** , **Lire** , **Verrouiller** , **CBA** et **RBA** dans la classe d'implémentation de comportement pour l'entité Voyage ainsi que dans la classe d'implémentation de comportement pour l'entité Réservation.

5.3 Création de la projection d'objet métier (Unité 3)

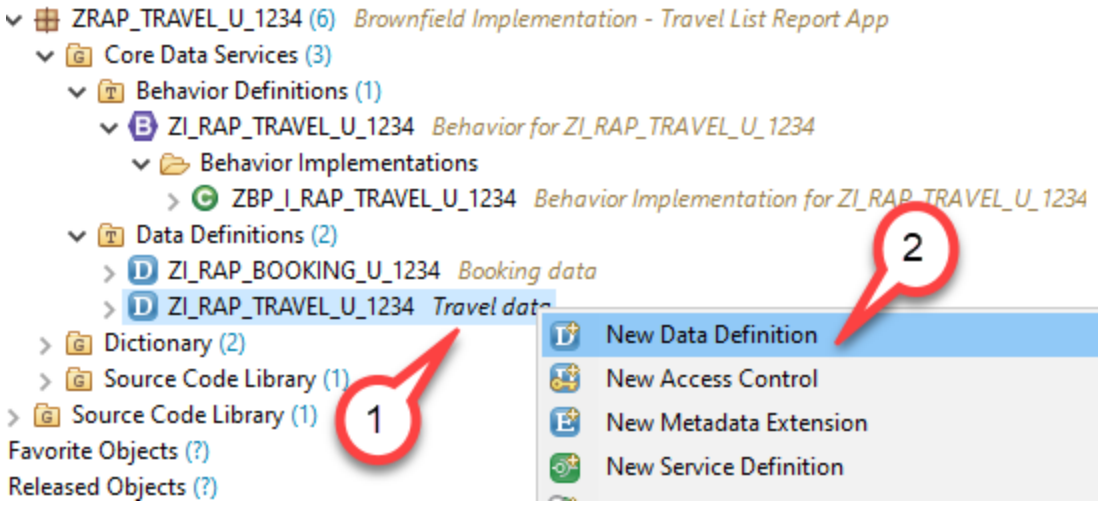
Objectif de l'unité

Créer la **projection** de l'objet métier ZI_RAP_Travel_U_#### pour adapter et exposer son comportement via un **service spécifique**, notamment en ajoutant des **annotations UI**, des **valeurs par défaut**, et en **préparant l'accès OData** pour une application SAP Fiori dans un scénario **unmanaged**.

Étape 1 : Créer une vue de projection racine pour Travel

Commençons par ajouter une couche de projection sur nos vues d'interface CDS.

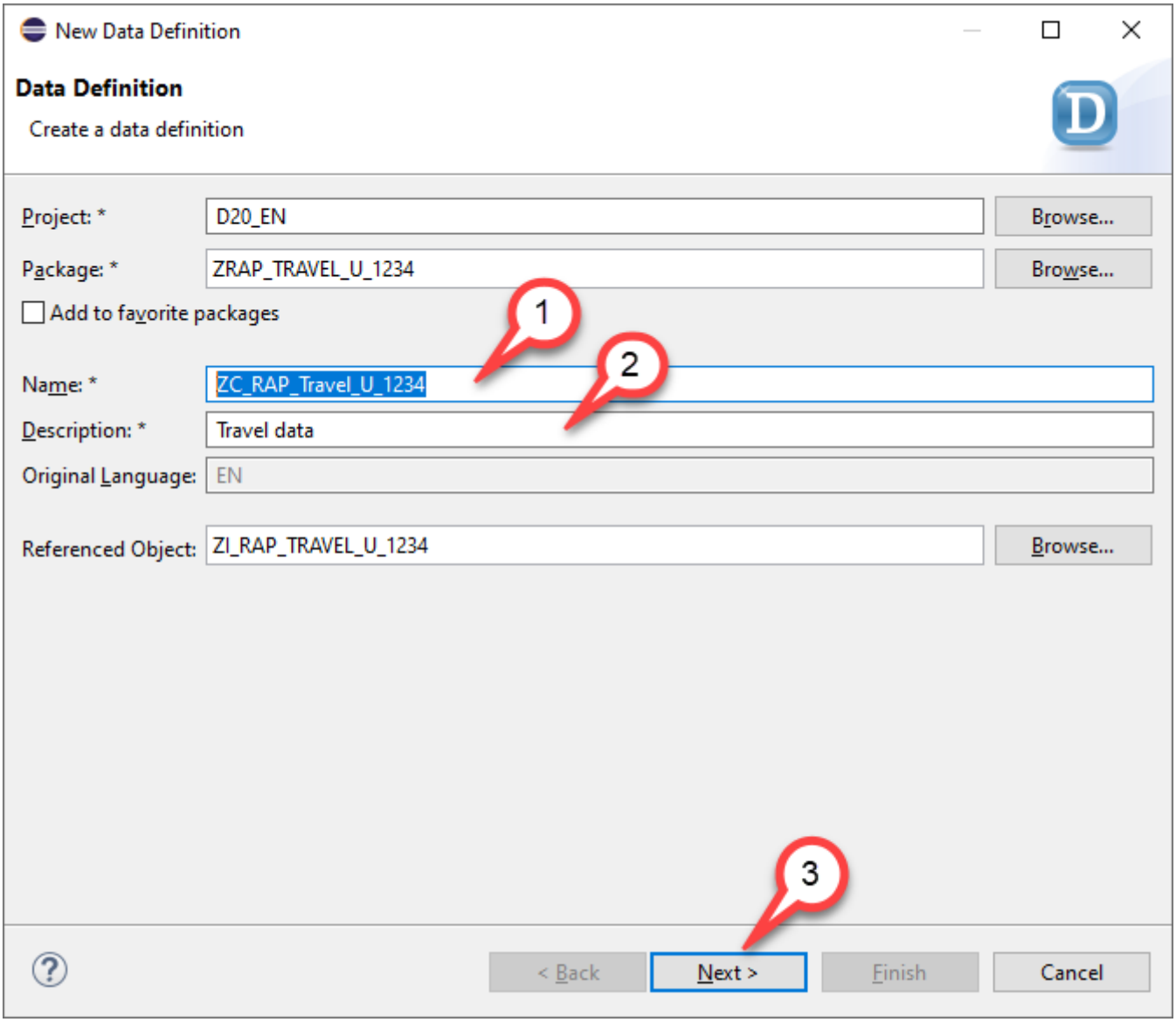
1. Nous pouvons cliquer avec le bouton droit sur la vue de l'interface CDS pour les voyages et utiliser le nouvel assistant pour créer des vues de projection qui ont notre vue d'interface CDS pour les voyages comme source de données.
    - Faites un clic droit sur la vue de l'interface ZI_RAP_Travel_U_####pour Voyager
    - Sélectionner **une nouvelle définition de données**



2. Dans l'assistant de définition de données
    - Nom : ZC_RAP_Travel_U_####
    - Description : Données de voyage

    Et appuyez sur **Suivant**

    La vue d'interface CDS ZI_RAP_Travel_U_#### est automatiquement définie comme **objet référencé**.

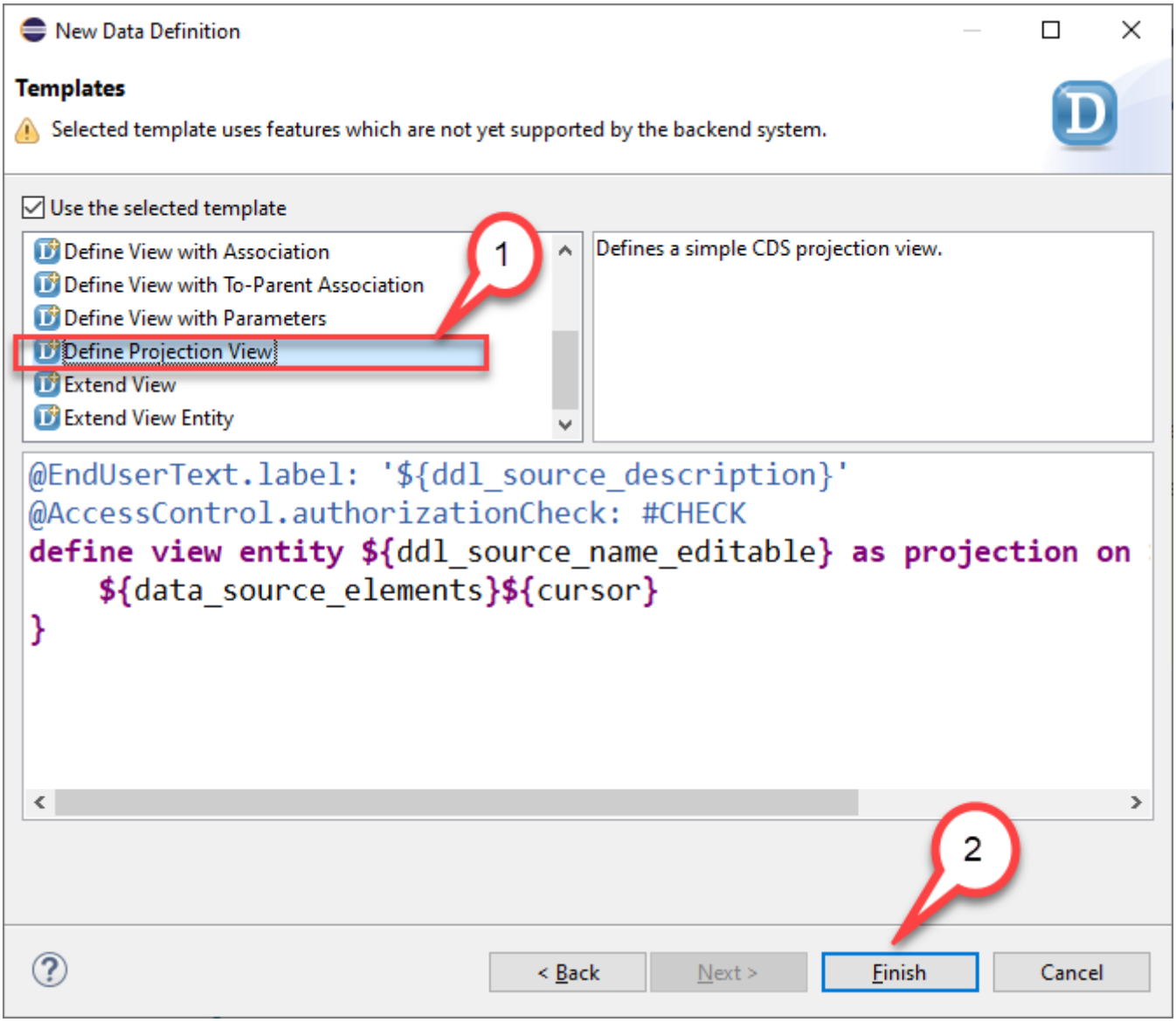


3. Sélection de la demande de transport

   - Sélectionnez une demande de transport
   - Appuyez **sur Suivant** mais **N'APPUYEZ PAS SUR TERMINER**

4. Choisissez le modèle

   - Assurez-vous de choisir le modèleDefine Projection View
   - Appuyez sur **Terminer**



5. Vérifier le code du modèle

   - Vous verrez l'avertissement *mot-clé ROOT manquant puisque ZI_RAP_TRAVEL_U_1234 a la propriété root*

```
Release Notes    *[D20] ZC_RAP_TRAVEL_U_1234
1  @EndUserText.label: 'Travel data'
2  @AccessControl.authorizationCheck: #CHECK
3  define view entity ZC_RAP_Travel_U_1234 as projection on ZI_RAP_T
4      //ZI_RAP_TRAVEL_U_1234
       key TravelID,
       AgencyID,
7      CustomerID,
8      BeginDate,
```

6. Ajoutez le mot-clé **root** à votre code afin que le début de la définition des données soit désormais

```
define root view entity ZC_RAP_Travel_U_#### as projection on
ZI_RAP_TRAVEL_U_####
```

```
Release Notes    *[D20] ZC_RAP_TRAVEL_U_1234
1  @EndUserText.label: 'Travel data'
2  @AccessControl.authorizationCheck: #CHECK
3  define root view entity ZC_RAP_Travel_U_1234 as projection on ZI_
4      //ZI_RAP_TRAVEL_U_1234
5      key TravelID,
6      AgencyID,
7      CustomerID,
8      BeginDate,
9      EndDate,
```

7. Rendre la vue de projection *consultable*

   - Ajoutez l'annotation @Search.searchable: true dans la section d'en-tête de la vue de projection
   - Créez les champs AgencyID et CustomerID les éléments de recherche par défaut en ajoutant l'annotation @Search.defaultSearchElement: true à ces champs.

```
Release Notes    [D20] ZC_RAP_TRAVEL_U_1234
1  @EndUserText.label: 'Travel da
2  @AccessControl.authorizationCheck: #CHECK
3  @Search.searchable: true
4  define root view entity ZC_RAP_Travel_U_1234
5    as projection on ZI_RAP_Travel_U_1234
6  {
7        //ZI_RAP_TRAVEL_U_1234
8    key TravelID,
9        @Search.defaultSearchElement: true
10       AgencyID,
11       @Search.defaultSearchElement: true
12       CustomerID,
13       BeginDate,
14       EndDate,
```

8. Ajouter de la valeur aide

   - Ajoutez l'annotation @Consumption.valueHelpDefinition: [{ entity: { name: '/DMO/I_Agency', element: 'AgencyID' } }]au champ AgencyID
   - Ajoutez l'annotation @Consumption.valueHelpDefinition: [{ entity: { name: '/DMO/I_Customer', element: 'CustomerID' } }]au champ CustomerID
   - Ajoutez l'annotation @Consumption.valueHelpDefinition: [{ entity: { name: 'I_Currency', element: 'Currency' } }]au champ CurrencyCode

```
key TravelID,
@Consumption.valueHelpDefinition: [ { entity: { name: '/DMO/I_Agency', element: 'AgencyID' } } ]
@Search.defaultSearchElement: true
AgencyID,
@Consumption.valueHelpDefinition: [ { entity: { name: '/DMO/I_Customer', element: 'CustomerID' } } ]
@Search.defaultSearchElement: true
CustomerID,
BeginDate,
EndDate,
BookingFee,
TotalPrice,
@Consumption.valueHelpDefinition: [ { entity: { name: 'I_Currency', element: 'Currency' } } ]
CurrencyCode,
Description,
Status
```

9. Ajouter des informations à l'association _*Réservation*

   - Ajouter l'information à l'association _Booking que cette association fait partie d'un arbre de composition

```
_Booking : redirected to composition child ZC_RAP_Booking_U_####
```

```
22      Lastchangedby,
23      Lastchangedat,
24
25      /* Associations */
26      _Agency,
27      _Booking : redirected to composition cl
28      _Currency,
29      _Customer
30  }
```

Votre vue de projection de voyage devrait maintenant ressembler à ceci :

```
@EndUserText.label: 'Travel data'
```

```
    @AccessControl.authorizationCheck: #CHECK

    @Search.searchable: true


    define root view entity ZC_RAP_Travel_U_####

      as projection on ZI_RAP_Travel_U_####

    {

      //ZI_RAP_TRAVEL_U_####

      key TravelID,

          @Consumption.valueHelpDefinition: [ { entity: { name: '/DMO/I_Agency',
element: 'AgencyID' } } ]

          @Search.defaultSearchElement: true

          AgencyID,

          @Consumption.valueHelpDefinition: [ { entity: { name: '/DMO/I_Customer',
element: 'CustomerID' } } ]

          @Search.defaultSearchElement: true

          CustomerID,

          BeginDate,

          EndDate,

          @Semantics.amount.currencyCode: 'CurrencyCode'

          BookingFee,

          @Semantics.amount.currencyCode: 'CurrencyCode'

          TotalPrice,

          @Consumption.valueHelpDefinition: [ { entity: { name: 'I_Currency',
element: 'Currency' } } ]

          CurrencyCode,

          Description,

          Status,

          Createdby,
```

```
        Createdat,

        Lastchangedby,

        Lastchangedat,


        /* Associations */

        //ZI_RAP_TRAVEL_U_####

        _Agency,

        _Booking : redirected to composition child ZC_RAP_Booking_U_####,

        _Currency,

        _Customer

    }
```

10. Enregistrez votre vue de projection de voyage.

Étape 2 : Créer une vue de projection pour la réservation

Nous créons maintenant une vue de projection pour notre entité enfant Booking.

1. Nous pouvons cliquer avec le bouton droit sur la vue de l'interface CDS pour **la réservation** et utiliser le nouvel assistant pour créer des vues de projection qui ont notre vue d'interface CDS pour la réservation comme source de données.

    - Faites un clic droit sur la vue de l'interface CDSZI_RAP_Booking_U_####
    - Sélectionner **une nouvelle définition de données**

2. Dans l'assistant de définition de données

    - Nom : ZC_RAP_BOOKING_U_####
    - Description : Données de réservation

    Et appuyez sur **Suivant**

    La vue d'interface CDS ZI_RAP_Booking_U_#### est automatiquement définie comme **objet référencé**.

3. Sélection de la demande de transport
   - Sélectionnez une demande de transport
   - Appuyez **sur Suivant** mais **N'APPUYEZ PAS SUR TERMINER**
4. Choisissez le modèle
   - Assurez-vous de choisir le modèleDefine Projection View
   - Appuyez sur **Terminer**
5. Rendre également la vue de projection enfant *consultable*
   - Ajoutez l'annotation @Search.searchable: truedans la section d'en-tête de la vue de projection enfant
   - Créez les champs TravelIDet BookingIDles éléments de recherche par défaut en ajoutant l'annotation @Search.defaultSearchElement: trueà ces champs.

Release Notes | [D20] ZC_RAP_TRAVEL_U_1234 | *[D20] ZC_RAP_BOOKING_U_1234

```
1  @EndUserText.label: 'Booking data'
2  @AccessControl.authorizationCheck: #CHECK
3  @Search.searchable: true
4  define view entity ZC_RAP_Booking_U_1234
5    as projection on ZI_RAP_Booking_U_1234
6  {
7        //ZI_RAP_BOOKING_U_1234
8        @Search.defaultSearchElement: true
9    key TravelID,
10       @Search.defaultSearchElement: true
11   key BookingID,
12       BookingDate,
13       CustomerID,
```

6. Ajoutez maintenant des annotations d'aide à la valeur pour les champs suivants dans la vue de projection de réservation
   - ID client

```
@Consumption.valueHelpDefinition: [ { entity: { name:     '/DMO/I_Customer',
```

```
                                                    element:     'CustomerID' }
} ]
```

- ID de l'opérateur

```
@Consumption.valueHelpDefinition: [ { entity: { name:     '/DMO/I_Carrier',
```

- ID de connexion

```
@Consumption.valueHelpDefinition: [ { entity: { name:    '/DMO/I_Flight',
                                                    element: 'ConnectionID' },
                                        additionalBinding: [ { localElement:
'FlightDate',
                                                                  element:
'FlightDate',
                                                                  usage: #RESULT },
                                                               { localElement:
'CarrierID',
                                                                     element:
'AirlineID',
                                                                       usage:
#RESULT },
                                                               { localElement:
'FlightPrice',
                                                                     element:
'Price',
                                                                       usage:
#RESULT },
                                                               { localElement:
'CurrencyCode',
                                                                     element:
'CurrencyCode',
                                                                       usage:
#RESULT } ]
                                        } ]
```

- Code de devise

```
@Consumption.valueHelpDefinition: [ {entity: { name:    'I_Currency',
                                                element: 'Currency' } } ]
```

7. Nous devons enfin ajouter des informations à l'association _Travelindiquant qu'elle fait partie d'un arbre de composition et qu'elle pointe vers l'entité parente Travel. _Travel : redirected to parent ZC_RAP_Travel_U_####

```
Release Notes   [D20] ZC_RAP_TRAVEL_U_1234   *[D20] ZC_RAP_BOOKING_U_1234
1  @EndUserText.label: 'Booking data'
2  @AccessControl.authorizationCheck: #CHECK
3  @Search.searchable: true
4
5
6  define view entity ZC_RAP_Booking_U_1234
7    as projection on ZI_RAP_Booking_U_1234
8  {
9        //ZI_RAP_BOOKING_U_####
10       @Search.defaultSearchElement: true
11   key TravelID,
12       @Search.defaultSearchElement: true
13   key BookingID,
14       BookingDate,
15       @Consumption.valueHelpDefinition: [ { entity: { name:     '/DMO/I_Customer',
16                                                      element:    'CustomerID' } } ]
17       CustomerID,
18       @Consumption.valueHelpDefinition: [ { entity: { name:     '/DMO/I_Carrier',
19                                                      element:    'AirlineID' } } ]
20       CarrierID,
21       @Consumption.valueHelpDefinition: [ { entity: { name:    '/DMO/I_Flight',
22                                                        element: 'ConnectionID' },
23                                             additionalBinding: [ { localElement: 'FlightDate',
24                                                                    element:      'FlightDate',
25                                                                    usage: #RESULT },
26                                                                  { localElement: 'CarrierID',
27                                                                        element: 'AirlineID',
28                                                                          usage: #RESULT },
29                                                                  { localElement: 'FlightPrice',
30                                                                        element: 'Price',
31                                                                          usage: #RESULT },
32                                                                  { localElement: 'CurrencyCode',
33                                                                        element: 'CurrencyCode',
34                                                                          usage: #RESULT } ]
35                                             } ]
36       ConnectionID,
37       FlightDate,
38       FlightPrice,
39       @Consumption.valueHelpDefinition: [ {entity: { name:    'I_Currency',
40                                                      element: 'Currency' } } ]
41       CurrencyCode,
42
43       /* Associations */
44       _Carrier,
45       _Customer,
46       _Travel : redirected to parent ZC_RAP_Travel_U_1234
47 }
```

8. Vérifiez le code

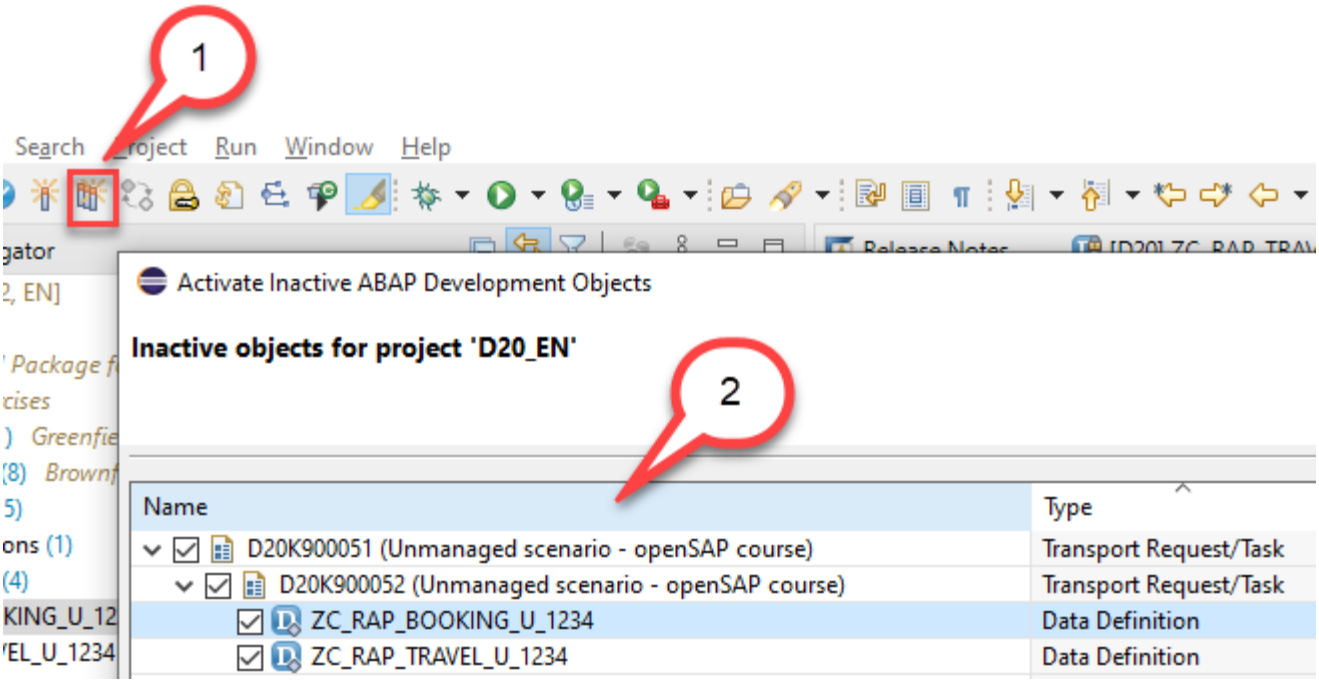
   - Ainsi, le code ressemble désormais à ceci :

```
@EndUserText.label: 'Booking'

  @AccessControl.authorizationCheck: #CHECK

  @Search.searchable: true
```

```
    define view entity ZC_RAP_Booking_U_####

      as projection on ZI_RAP_Booking_U_####

    {

          //ZI_RAP_BOOKING_U_####

          @Search.defaultSearchElement: true

      key TravelID,

          @Search.defaultSearchElement: true

      key BookingID,

          BookingDate,

          @Consumption.valueHelpDefinition: [ { entity: { name:
'/DMO/I_Customer',

                                                        element:     'CustomerID' }
} ]

          CustomerID,

          @Consumption.valueHelpDefinition: [ { entity: { name:
'/DMO/I_Carrier',

                                                        element:     'AirlineID' }
} ]

          CarrierID,

          @Consumption.valueHelpDefinition: [ { entity: { name:
'/DMO/I_Flight',

                                                           element: 'ConnectionID'
},

                                                  additionalBinding: [ {
localElement: 'FlightDate',

                                                                          element:
'FlightDate',

                                                                          usage:
#RESULT },
```

```
                                                                    {
localElement: 'CarrierID',

element: 'AirlineID',

usage: #RESULT },
                                                                    {
localElement: 'FlightPrice',

element: 'Price',

usage: #RESULT },
                                                                    {
localElement: 'CurrencyCode',

element: 'CurrencyCode',

usage: #RESULT } ]
                                                  } ]
          ConnectionID,
          FlightDate,
          FlightPrice,

          @Consumption.valueHelpDefinition: [ {entity: { name:    'I_Currency',
                                                          element: 'Currency' } } ]
          CurrencyCode,
          /* Associations */
          _Carrier,
          _Customer,
          _Travel : redirected to parent ZC_RAP_Travel_U_####
    }
```

9. Nous pouvons maintenant activer à la fois la vue de projection pour les données de voyage ainsi que celle pour les données de réservation.
    - Dans le menu, choisissez l'icône d'activation **de masse** ou appuyez sur **Ctrl+Maj+F3**
    - Dans la troisième boîte de dialogue, assurez-vous que tous les objets sont sélectionnés



Étape 3 : Créer une vue d'extension de métadonnées pour la projection de voyage

Si vous souhaitez implémenter un service de consommation basé sur l'interface utilisateur, les extensions de métadonnées font partie de la couche de projection. Ces extensions contiennent des annotations spécifiques à l'interface utilisateur et ont été conçues pour séparer les annotations basées sur l'interface utilisateur des vues de projection. Elles permettent également une superposition, permettant aux secteurs d'activité, aux partenaires ou aux clients d'ajouter leurs annotations à celles existantes.

1. Cliquez avec le bouton droit sur notre vue de projection nouvellement créée pour les voyages ZC_RAP_Travel_U_#### et sélectionnez **Nouvelle extension de métadonnées**.

2. Spécifiez les valeurs suivantes
    - Nom:ZC_RAP_Travel_U_####
    - Description:Travel data

    Appuyez **sur Suivant**

3. Sélection de la demande de transport
   - Sélectionnez une demande de transport
   - Appuyez **sur Suivant** mais **N'APPUYEZ PAS SUR TERMINER**
4. Choisissez un modèle
   - Choisir Annotate view
   - Appuyez sur **Terminer**
5. Modifier le modèle
   - En tant que couche de métadonnées, nous choisissons #CORE.
   - Nous ajoutons des annotations @UI.headerInfo pour l'en-tête et @UI.presentationVariant: [{ }] pour le presentation Variant, qui définit l'ordre de tri par défaut.
   - Nous ajoutons l'annotation @UI.hidden: true aux champs administratifs, afin que les champs administratifs soient masqués sur l'interface utilisateur
   - Nous ajoutons une annotation d'identification de ligne afin que les champs soient affichés dans la liste et la page d'objet. Par exemple

```
@UI: { lineItem:        [ { position: 50 } ],

       identification:  [ { position: 50 } ]  }
```

Votre code devrait donc maintenant ressembler à ce qui suit

```
@Metadata.layer: #CORE

@UI: {

  headerInfo: {

    typeName: 'Travel',

    typeNamePlural: 'Travels',

    title: {

      type: #STANDARD,

      label: 'Travel',

      value: 'TravelID'

    }
```

```
  },
  presentationVariant: [ { sortOrder: [ { by:         'TravelID',
                                          direction:  #DESC }] }]
}
annotate view ZC_RAP_Travel_U_#### with
{
  @UI.facet: [ { id:            'Travel',
                 purpose:       #STANDARD,
                 type:          #IDENTIFICATION_REFERENCE,
                 label:         'Travel',
                 position:      10 },
               { id:            'Booking',
                 purpose:       #STANDARD,
                 type:          #LINEITEM_REFERENCE,
                 label:         'Booking',
                 position:      20,
                 targetElement: '_Booking'}]

  @UI: { lineItem:        [ { position: 10 } ],
         identification:  [ { position: 10 } ]  }
  TravelID;

  @UI: { lineItem:        [ { position: 20 } ],
         identification:  [ { position: 20 } ]  }
  AgencyID;
```

```
@UI: { lineItem:        [ { position: 30 } ],
       identification:  [ { position: 30 } ]  }
CustomerID;

@UI: { lineItem:        [ { position: 40 } ],
       identification:  [ { position: 40 } ]  }
BeginDate;

@UI: { lineItem:        [ { position: 50 } ],
       identification:  [ { position: 50 } ]  }
EndDate;

@UI: { lineItem:        [ { position: 60 } ],
       identification:  [ { position: 60 } ]  }
BookingFee;

@UI: { lineItem:        [ { position: 70 } ],
       identification:  [ { position: 70 } ]  }
TotalPrice;

@UI: { identification:  [ { position: 80 } ]  }
Description;

@UI: { lineItem:        [ { position: 90 } ],
       identification:  [ { position: 90 } ]  }
Status;
```

```
  @UI.hidden: true
  Createdby;

  @UI.hidden: true
  Createdat;

  @UI.hidden: true
  Lastchangedby;

  @UI.hidden: true
  Lastchangedat;

}
```

6. Cependant, nous constatons toujours une erreur indiquant que l'annotation @Metadata.allowExtensionsest manquante dans la vue de projection qui doit être annotée.

Release Notes | [D20] ZC_RAP_TRAVEL_U_1234 | [D20] ZC_RAP_BOOKING_U_1234 | [D20] ZC_RAP_TRAVEL_U_1234

```
1  @Metadata.layer: #CORE
2  @UI: {
3    headerInfo: {
4      typeName: 'Travel',
5      typeNamePlural: 'Travels',
6      title: {
7        type: #STANDARD,
8        label: 'Travel',
9        value: 'TravelID'
10     }
11   },
12   presentationVariant: [ { sortOrder: [ { by:         'TravelID',
13                                          direction:  #DESC }] }]
14 }
15 annotate view ZC_RAP_Travel_U_1234 with
16 {
17   @UI.facet: [ { id:             'Travel',
18                  purpose:        #STANDARD,
19                  type:           #IDENTIFICATION_REFERENCE,
20                  label:          'Travel',
21                  position:       10 },
22                { id:             'Booking',
23                  purpose:        #STANDARD,
24                  type:           #LINEITEM_REFERENCE,
25                  label:          'Booking',
26                  position:       20,
27                  targetElement:  '_Booking'}]
28
29   @UI: { lineItem:        [ { position: 10 } ],
30          identification:  [ { position: 10 } ]  }
31   TravelID;
32
33   @UI: { lineItem:        [ { position: 20 } ],
34          identification:  [ { position: 20 } ]  }
35   AgencyID;
36
37   @UI: { lineItem:        [ { position: 30 } ],
38          identification:  [ { position: 30 } ]  }
39   CustomerID;
40
41   @UI: { lineItem:        [ { position: 40 } ],
42          identification:  [ { position: 40 } ]  }
43   BeginDate;
44
45   @UI: { lineItem:        [ { position: 50 } ],
46          identification:  [ { position: 50 } ]  }
```

7. Ici, nous pouvons simplement appuyer sur **Ctrl+1** et utiliser le correctif rapide qui ajoutera cette annotation à la vue de base.
   - Sélectionnez le nom de la vue de projection ZC_RAP_Travel_U_####
   - Appuyez sur **Ctrl+1**
   - Double-cliquez sur le correctif rapide Add annoation 'Metadata.allowextension' to base view

```
15 annotate view ZC_RAP_Travel_U_1234 with
16 {
17   @UI.facet: [ { id:
18                  purpose:
19                  type:
20                  label:
21                  position
22                { id:
23                  purpose:
24                  type:
25                  label:
26                  position
27                  targetEl
28
29   @UI: { lineItem:
30          identification:
31   TravelID;
32
```

Add annotation 'Metadata.allowExtensions' to base view

Press 'Ctrl+Shift+1' to show in Quick Assist View

8. Nous activons les modifications dans la vue de projection

```
Release Notes | *[D20] ZC_RAP_TRAVEL_U_1234
1 @EndUserText.label: 'Travel data'
2 @AccessControl.authorizationCheck: #CHECK
3 @Search.searchable: true
4 @Metadata.allowExtensions: true
5 define root view entity ZC_RAP_Travel_U_1234
6   as projection on ZI_RAP_Travel_U_1234
7 {
```

9. Nous pouvons maintenant revenir à notre vue d'extension de métadonnées et l'activer.

```
Release Notes | [D20] ZC_RAP_TRAVEL_U_1234 | [D20] ZC_RAP_BOOKING_U_1234 | [D20] ZC_RAP_TRAVEL_U_1234
1 @Metadata.layer: #CORE
2 @UI: {
3   headerInfo: {
4     typeName: 'Travel',
5     typeNamePlural: 'Travels',
6     title: {
7       type: #STANDARD,
8       label: 'Travel',
9       value: 'TravelID'
10    }
11  },
12  presentationVariant: [ { sortOrder: [ { by:        'TravelID',
13                                          direction:  #DESC }] }]
14 }
15 annotate view ZC_RAP_Travel_U_1234 with
16 {
17   @UI.facet: [ { id:          'Travel',
18                  purpose:     #STANDARD,
19                  type:        #IDENTIFICATION_REFERENCE,
20                  label:       'Travel',
```

Étape 4 : Créer une vue d'extension de métadonnées pour la réservation

Nous pouvons maintenant faire la même chose pour la vue de projection de réservation et créer une vue d'extension de métadonnées pour celle-ci.

1. Cliquez avec le bouton droit sur notre vue de projection nouvellement créée pour *la réservation* ZC_RAP_Booking_U_#### et sélectionnez **Nouvelle extension de métadonnées**.

2. Spécifiez les valeurs suivantes

   - Nom:ZC_RAP_Booking_U_####
   - Description:Booking data

   Appuyez **sur Suivant**

3. Sélection de la demande de transport

   - Sélectionnez une demande de transport
   - Appuyez **sur Suivant** mais **N'APPUYEZ PAS SUR TERMINER**

4. Choisissez un modèle

   - ChoisirAnnotate view

   - Appuyez sur **Terminer**

5. Modifier le modèle

   - Nous choisissons à nouveau la couche de métadonnées #CORE.
   - Nous ajoutons des annotations @UI.headerInfopour l'en-tête et nous ajoutons l'annotation @UI.hidden: true cette fois uniquement pour le champ clé TravelID
   - Nous ajoutons @UI.lineItemdes @UI.identificationannotations pour que les champs soient affichés dans la liste et sur la page de l'objet. Par exemple

```
@UI: { lineItem:        [ { position: 50 } ],

       identification:  [ { position: 50 } ]  }
```

Votre code devrait maintenant ressembler à ce qui suit

```
@Metadata.layer: #CORE

    @UI: {

      headerInfo: {

        typeName: 'Booking',
```

```
      typeNamePlural: 'Bookings',
      title: {
        type: #STANDARD,
        label: 'Booking',
        value: 'BookingID'
      }
    }
  }
  annotate view ZC_RAP_Booking_U_#### with
  {
    @UI.facet: [ { id:            'Booking',
                   purpose:       #STANDARD,
                   type:          #IDENTIFICATION_REFERENCE,
                   label:         'Booking',
                   position:      10 } ]

    @UI.hidden: true
    TravelID;

    @UI: { lineItem:        [ { position: 10 } ],
           identification:  [ { position: 10 } ]  }
    BookingID;

    @UI: { lineItem:        [ { position: 20 } ],
           identification:  [ { position: 20 } ]  }
    BookingDate;
```

```
  @UI: { lineItem:        [ { position: 30 } ],
         identification:  [ { position: 30 } ]  }
  CustomerID;

  @UI: { lineItem:        [ { position: 40 } ],
         identification:  [ { position: 40 } ]  }
  CarrierID;

  @UI: { lineItem:        [ { position: 50 } ],
         identification:  [ { position: 50 } ]  }
  ConnectionID;

  @UI: { lineItem:        [ { position: 60 } ],
         identification:  [ { position: 60 } ]  }
  FlightDate;

  @UI: { lineItem:        [ { position: 70 } ],
         identification:  [ { position: 70 } ]  }
  FlightPrice;
}
```

6. Encore une fois, nous voyons une erreur indiquant que l'annotation @Metadata.allowExtensionsest manquante dans la vue de projection ZC_RAP_Booking_U_####qui doit être annotée.

```
Release Notes    [D20] ZC_RAP_TRAVEL_U_1234    [D20] ZC_RAP_BOOKIN
1  @Metadata.layer: #CORE
2  @UI: {
3    headerInfo: {
4      typeName: 'Booking',
5      typeNamePlural: 'Bookings',
6      title: {
7        type: #STANDARD,
8        label: 'Booking',
9        value: 'BookingID'
10     }
11   }
12 }
13 annotate view ZC_RAP_Booking_U_1234 with
14 {
15   @UI.facet: [ { id:              'Booking',
16                  purpose:        #STANDARD,
17                  type:           #IDENTIFICATION_REFERENCE,
18                  label:          'Booking',
19                  position:       10 } ]
20
21   @UI.hidden: true
22   TravelID;
23
24   @UI: { lineItem:        [ { position: 10 } ],
```

Annotation 'Metadata.allowExtensions' missing in 'ZC_RAP_Booking_U_1234'

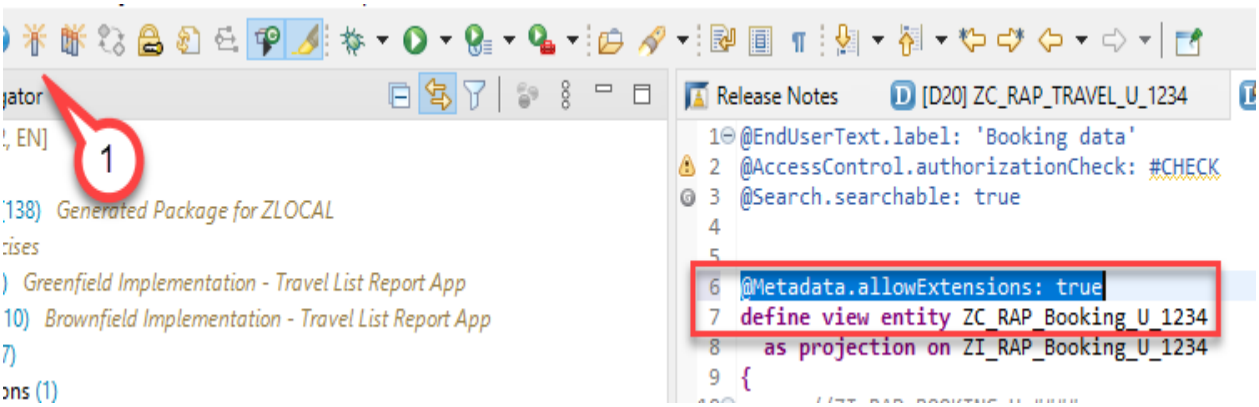
7. Encore une fois, nous pouvons simplement appuyer sur Ctrl+1 et utiliser le correctif rapide qui ajoutera cette annotation à la vue de base.
   - Sélectionnez le nom de la vue de projection ZC_RAP_Booking_U_####
   - Appuyez sur **Ctrl+1**
   - Double-cliquez sur le correctif rapide Add annoation 'Metadata.allowextension' to base view

```
1  @Metadata.layer: #CORE
2  @UI: {
3    headerInfo: {
4      typeName: 'Booking',
5      typeNamePlural: 'Bookings',
6      title: {
7        type: #STANDARD,
8        label: 'Booking',
9        value: 'BookingID'
10     }
11   }
12 }
13 annotate view ZC_RAP_Booking_U_1234 with
14 {
15   @UI.facet: [
16
17
18
19
20
21   @UI.hidden: t
22   TravelID;
23
24   @UI: { lineIt
25          identi
26   BookingID;
27
28   @UI: { lineIt
29          identi
30   BookingDate;
```

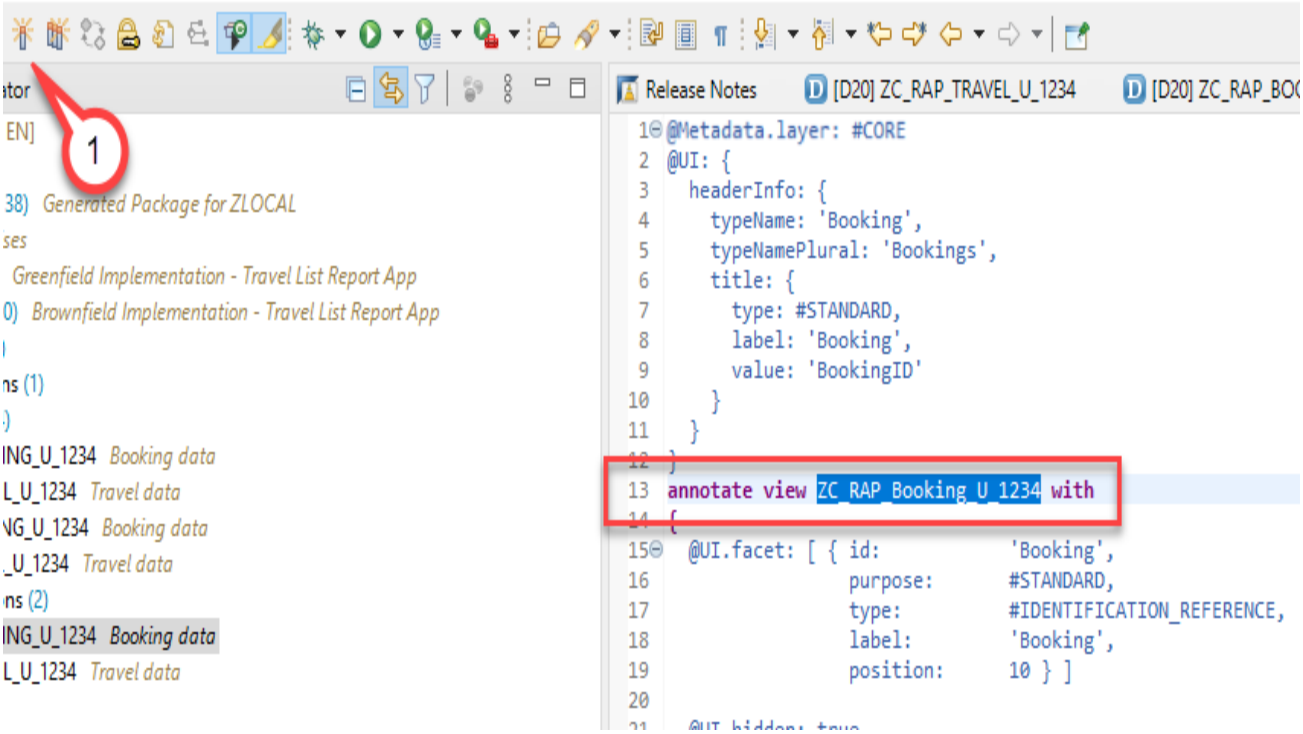
Add annotation 'Metadata.allowExtensions' to base view

Press 'Ctrl+Shift+1' to show in Quick Assist View

8. Nous activons les modifications dans la vue de projection



9. Nous pouvons maintenant revenir à notre vue d'extension de métadonnées et l'activer.



Étape 5 : Créer une définition de comportement pour la vue de projection

Le dernier objet que nous devons créer sur la couche de projection est une projection pour notre définition de comportement.

1. Pour cela, nous faisons un clic droit sur la définition des données de déplacement de la couche de projection et choisissons **Nouvelle définition de comportement**.

2. Dans la boîte de dialogue **Créer une définition de comportement**, tout est prérempli, il suffit d'appuyer sur Suivant.

3. Sélection de la demande de transport

   - Sélectionnez une demande de transport
   - Appuyez sur **Terminer**

4. L'éditeur de définition de comportement s'ouvre

   - Le type d'implémentation projection est présélectionné.
   - Tout ce que nous avons à faire est d'ajouter un alias pour les deux entités et de définir l'utilisation des ETags pour les deux entités.

```
1  projection;
2
3  define behavior for ZC_RAP_Travel_U_1234 alias Travel
4  use etag
5  {
6    use create;
7    use update;
8    use delete;
9
10   use association _Booking { create; }
11 }
12
13 define behavior for ZC_RAP_Booking_U_1234 alias Booking
14 use etag
15 {
16   use update;
17   use delete;
18
19   use association _Travel;
20 }
```

Votre code devrait ressembler à ceci :

```
projection;



 define behavior for ZC_RAP_Travel_U_#### alias Travel

 use etag

 {

   use create;

   use update;
```

```
  use delete;


  use association _Booking { create; }
}


define behavior for ZC_RAP_Booking_U_#### alias Booking
use etag
{
  use update;
  use delete;


  use association _Travel;
}
```

5. Enregistrez et activez vos modifications

Résumé

Dans cette unité, vous avez appris à :

- Créer la **projection CDS** pour un objet métier unmanaged
- Ajouter des **annotations UI** (recherche, aide à la saisie, en-tête, facettes, variante de présentation)
- Marquer une **association** comme enfant de composition (_Booking)
- Créer une **projection de comportement**
- Activer les opérations standards (create, update, delete)
- Gérer la **concurrence optimiste** via l'annotation use etag

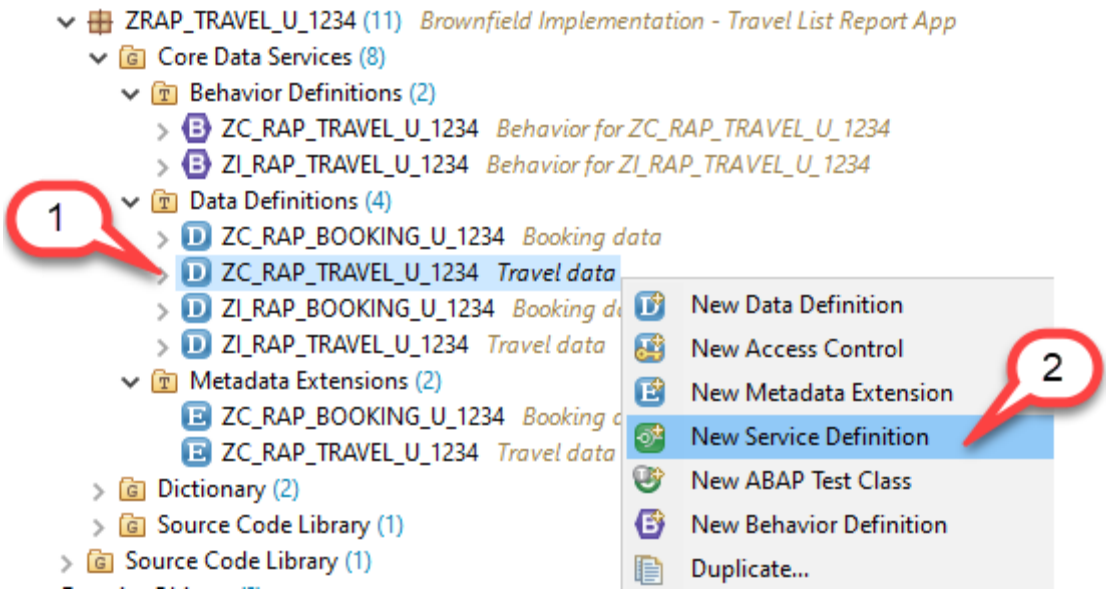
5.4 Création et prévisualisation du service d'interface utilisateur OData (Unité 4)

Objectif de l'unité

Créer la définition et la liaison de service **OData** pour exposer l'objet métier non géré via les projections **CDS**, et tester l'application **SAP Fiori Elements** en aperçu **UI**, avec gestion des alias pour les entités d'aide à la saisie.

Étape 1 : Créer une définition de service

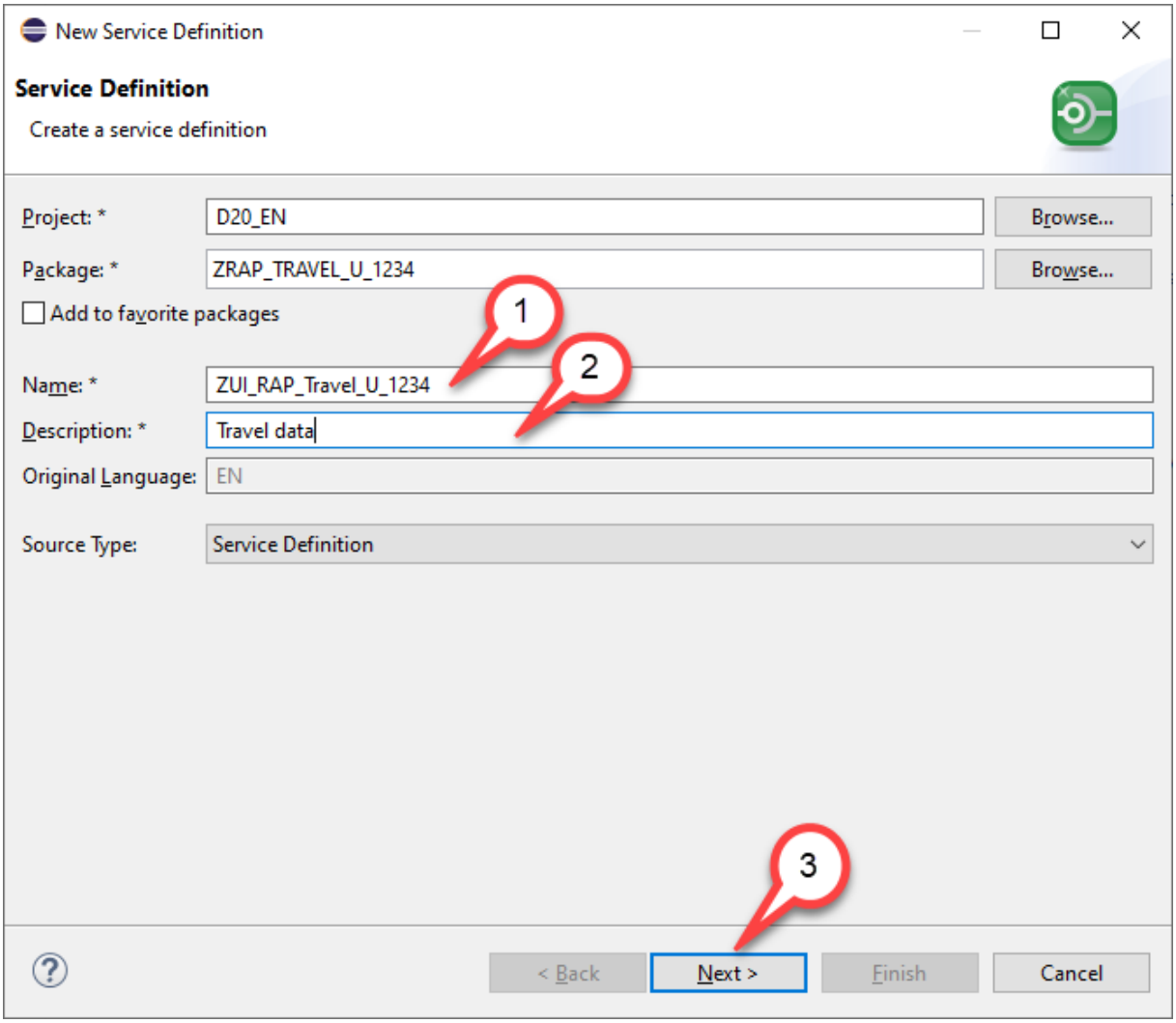
1. Cliquez avec le bouton droit sur la vue de projection des données de déplacement ZC_RAP_TRAVEL_U_####et sélectionnez **Nouvelle définition de service**



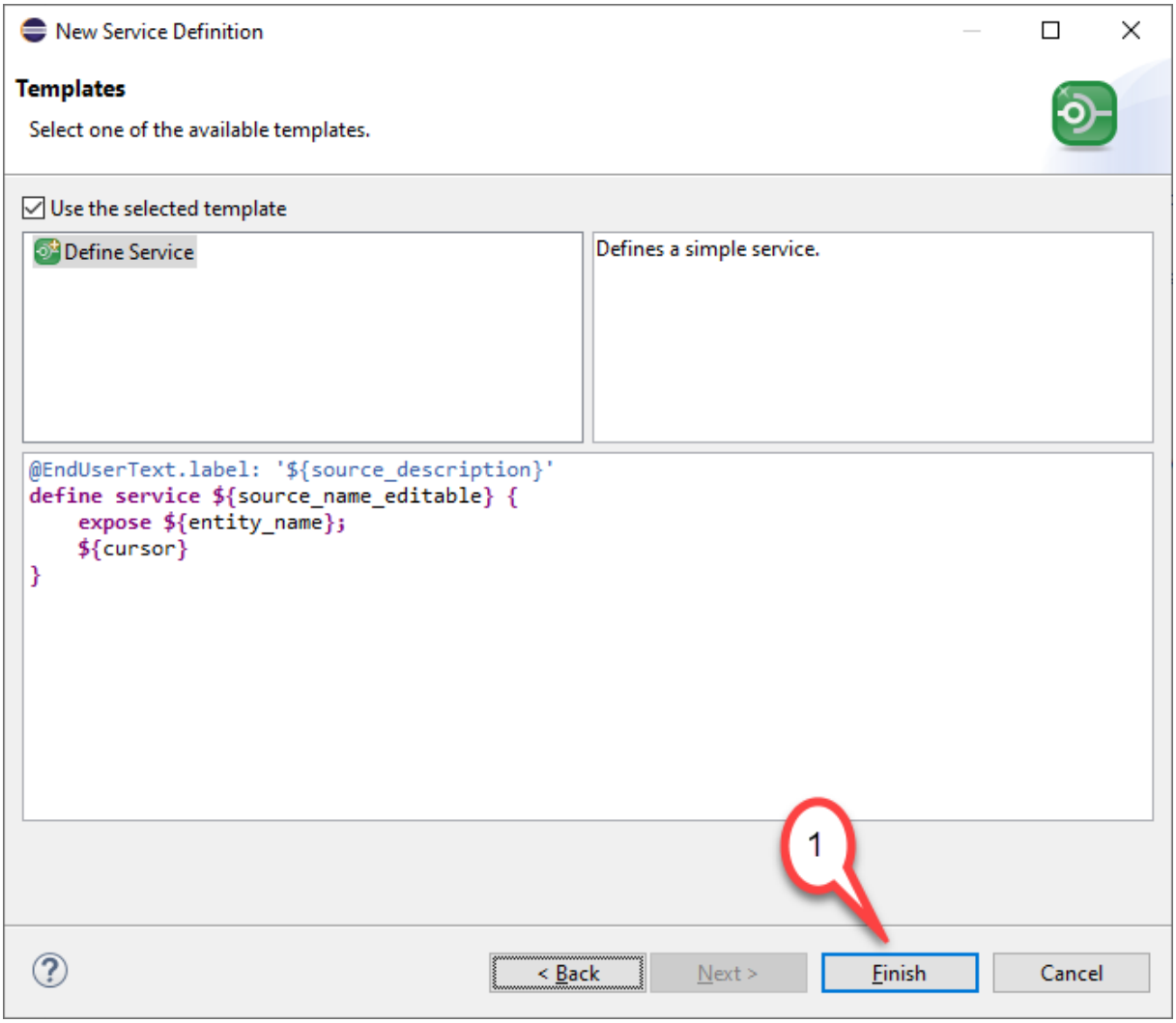
2. Saisissez les données suivantes
   - Nom:ZUI_RAP_Travel_U_####
   - Description:Travel data

Et appuyez sur **Suivant**

L'abréviation UI indique que cette définition de service est utilisée pour une consommation basée sur l'interface utilisateur. Pour une API Web, nous aurions utilisé **« API »** au lieu de **« UI »**. Notez qu'il est également possible d'ajouter la version du protocole OData au nom de la définition de service et à la liaison de service. Le nom de la définition de service serait alors « » ZUI_RAP_Travel_U_####_O2.



3. Sélection de la demande de transport
   - Sélectionnez une demande de transport
   - Appuyez **sur Suivant** mais **N'APPUYEZ PAS SUR TERMINER**
4. Nouvelle boîte de dialogue de définition de service
   - Comme le modèle Définir le service est déjà sélectionné, nous appuyons simplement sur **Terminer**



5. Comme indiqué précédemment, nous ajoutons, en plus des vues CDS faisant partie de l'arbre de composition, toutes les vues CDS utilisées comme aides à la valeur dans notre service. Le code de votre définition de service ZUI_RAP_Travel_U_####devrait maintenant ressembler à ceci :

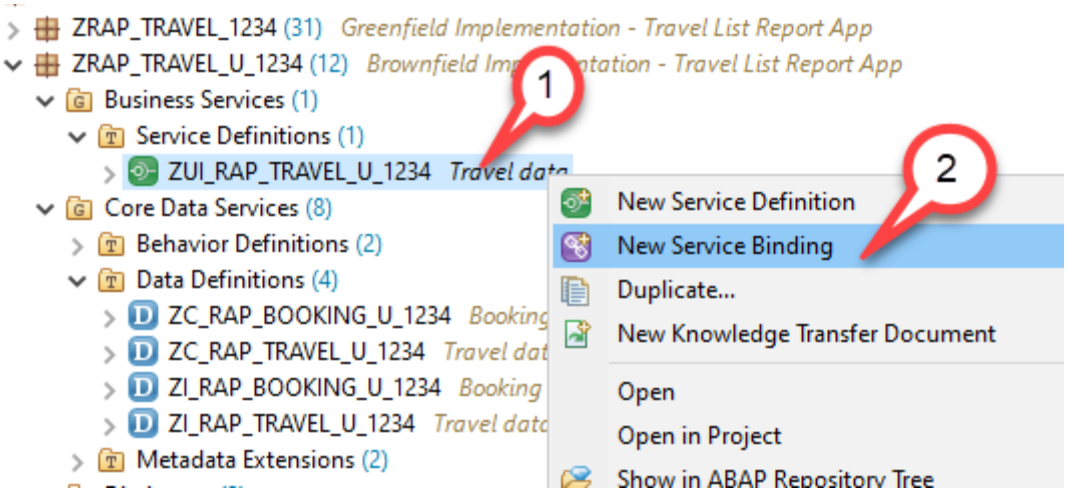
```
@EndUserText.label: 'Travel data'

define service ZUI_RAP_Travel_U_#### {

  expose ZC_RAP_Travel_U_#### as Travel;

  expose ZC_RAP_Booking_U_#### as Booking;

  expose /DMO/I_Customer as Passenger;

  expose /DMO/I_Agency as TravelAgency;
```

```
  expose /DMO/I_Carrier as Airline;

  expose /DMO/I_Flight as Flight;

  expose /DMO/I_Connection as Connection;

  expose /DMO/I_Airport as Airport;

  expose I_Currency as Currency;

}
```

6. Enregistrez et activez vos modifications

Étape 2 : Créer une liaison de service

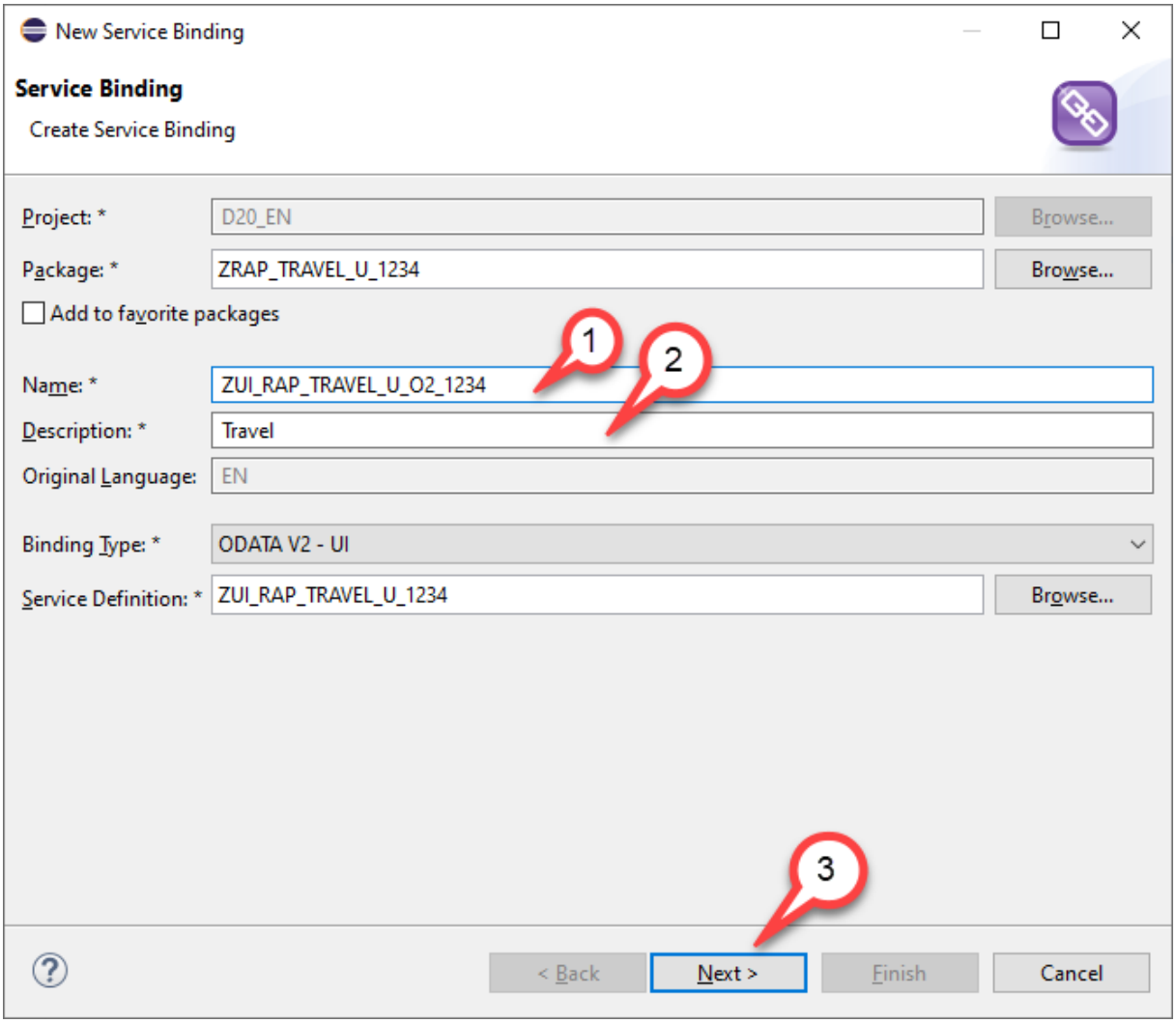
1. Nous faisons un clic droit sur la définition du service ZUI_RAP_Travel_U_####et choisissons **Nouvelle liaison de service** .



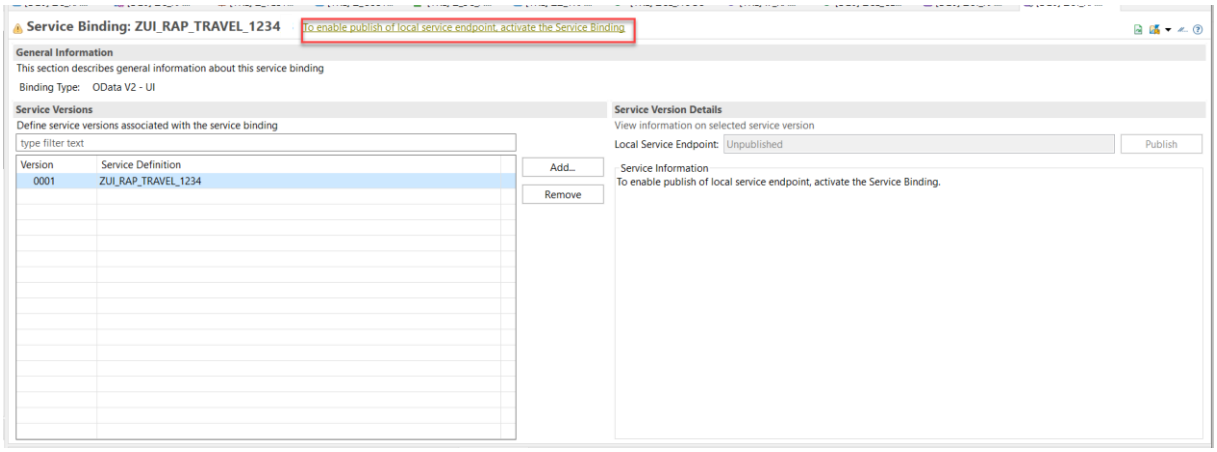
2. Dans la boîte de dialogue *Nouvelle liaison de service,* nous saisissons les valeurs suivantes
   - Nom: ZUI_RAP_TRAVEL_U_O2_####
   - Description:Travel

Laissez le type de liaison OData V2 - UIinchangé.

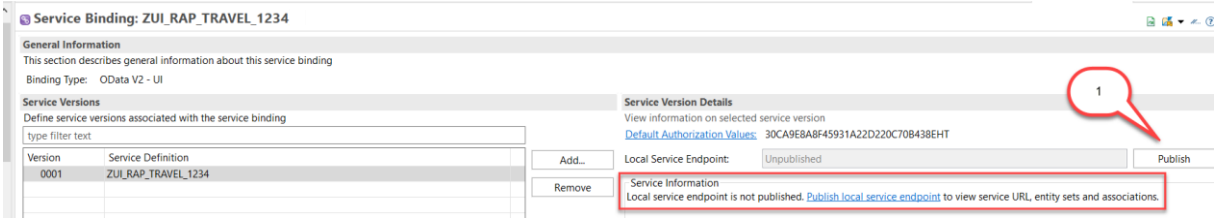
Et nous appuyons sur **Suivant**



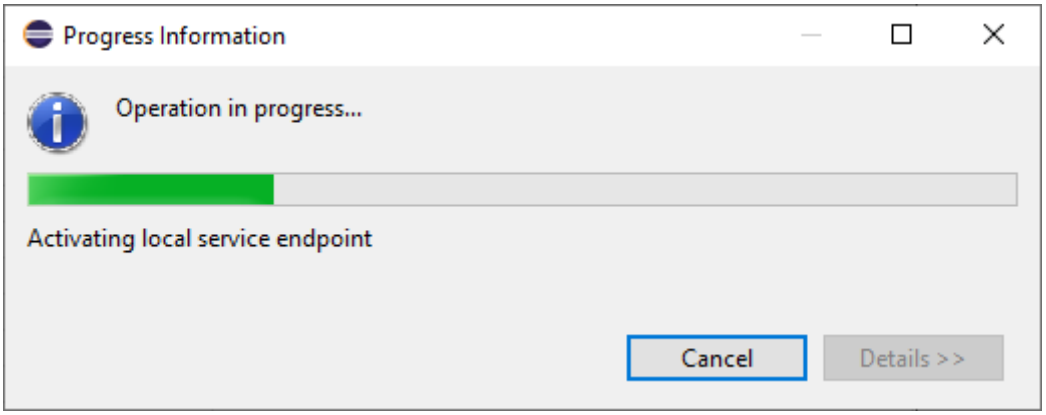
3. Sélection de la demande de transport
   - Sélectionnez une demande de transport
   - Appuyez sur **TERMINER**
4. La boîte de dialogue Service Binding s'ouvre et nous remarquons que le Service Binding doit être activé pour que nous puissions le tester.
   - Pressez



5. Publier la liaison de service
    - Appuyez sur **Publier**

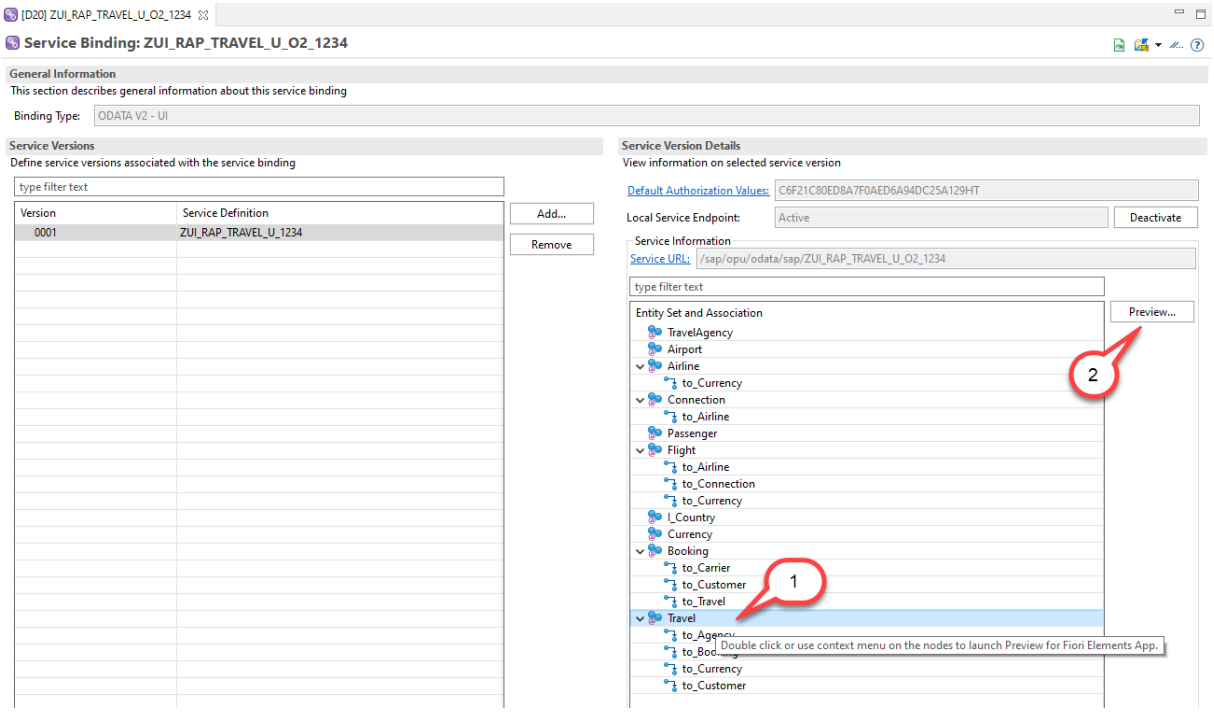


6. La publication du service prend un certain temps...

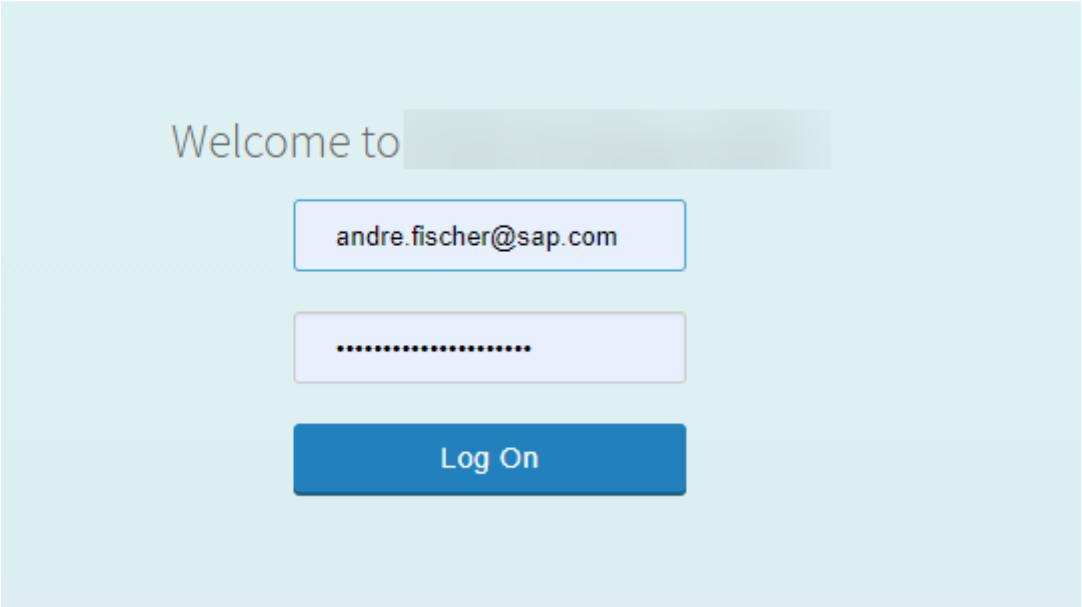


Une fois la liaison de service activée, nous pouvons voir tous les ensembles d'entités composant notre service. L'URL du service est également visible. En cliquant dessus, une fenêtre de navigateur s'ouvre et nous pouvons tester le document de service OData ou consulter le document $metadata.

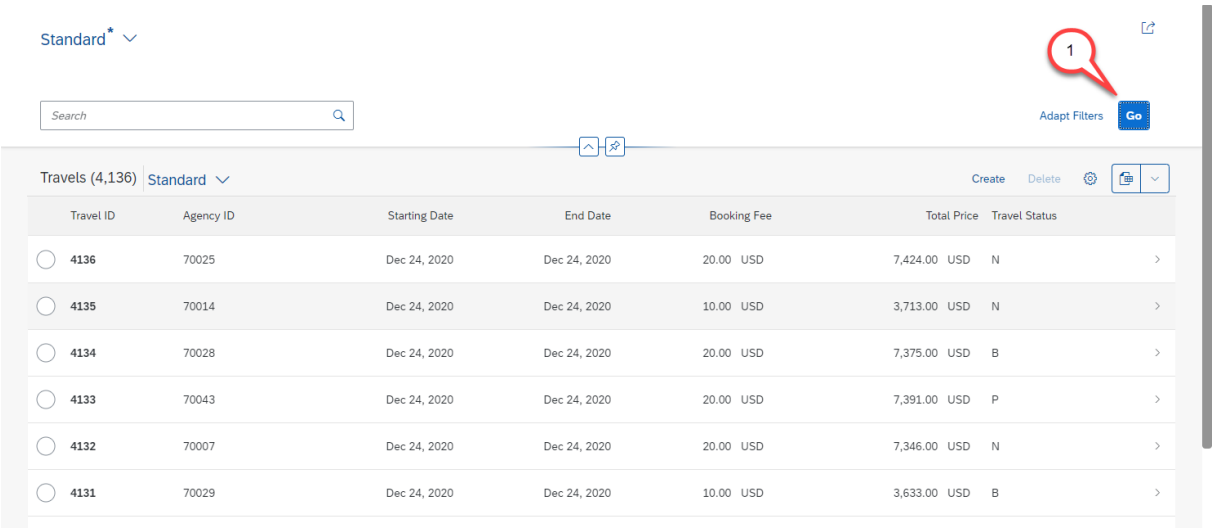
6. Tester le service en utilisant l'aperçu de Fiori Elements
    - Sélectionnez l'ensemble d'entités Travel
    - Cliquez sur **Aperçu**



7. Vous devez vous authentifier



8. Enfin, l'aperçu de Fiori Elements s'ouvre. En appuyant sur **« Go »**, l'application lit les données des anciennes tables.



Résumé

Dans cette unité, nous avons la liaison de service et avons publié notre service en créant une liaison de service.

6. Semaine 5 – Consommation de services et API Web

6.1 Création du modèle de consommation de services (Unité 1)

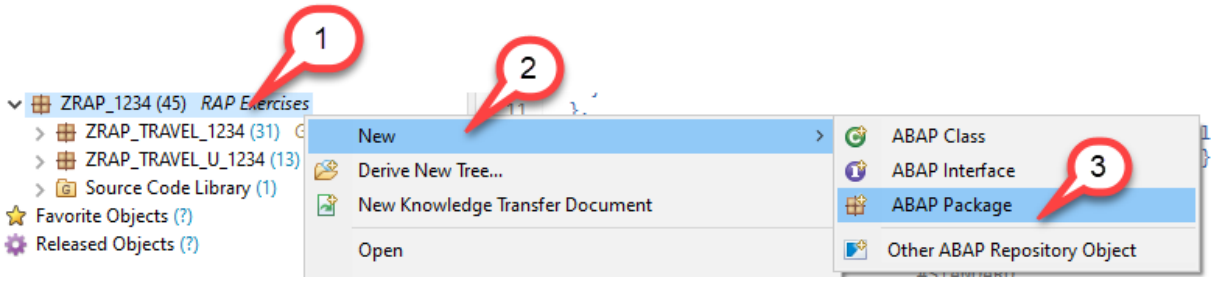
Objectif de l'unité

Créer un **modèle de consommation de services** pour intégrer un **service OData externe** (liste des agences) dans l'application, à l'aide de l'assistant SAP, qui génère la **définition de service**, l'**entité abstraite** et le **modèle de consommation**.

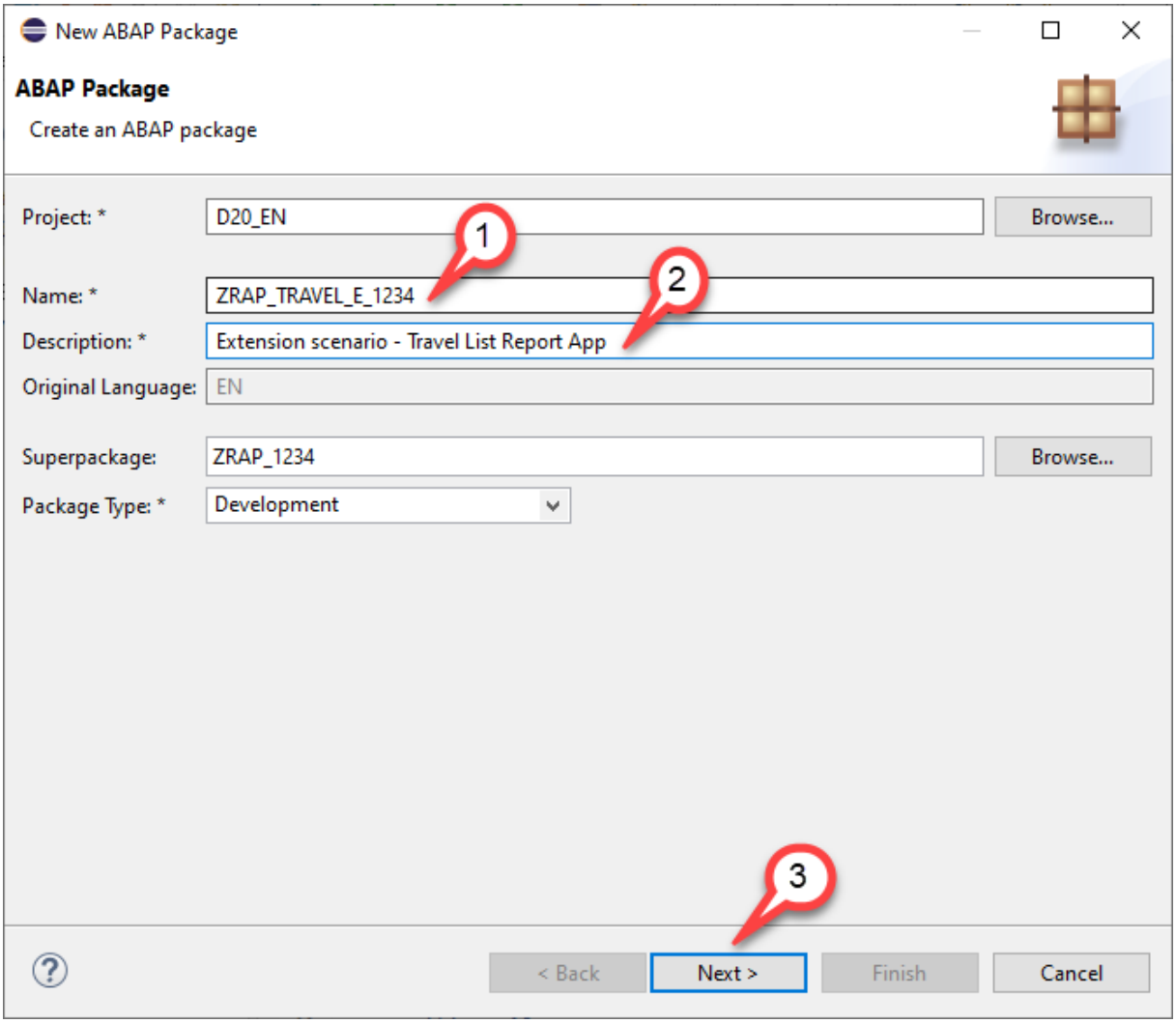
Étape 1 : Créer le package

Créez d'abord un nouveau package pour regrouper les différents artefacts de développement que vous allez créer pour étendre votre application de voyage gérée. Il s'agira d'un sous-package du package **ZRAP_####** créé lors de la semaine 1.

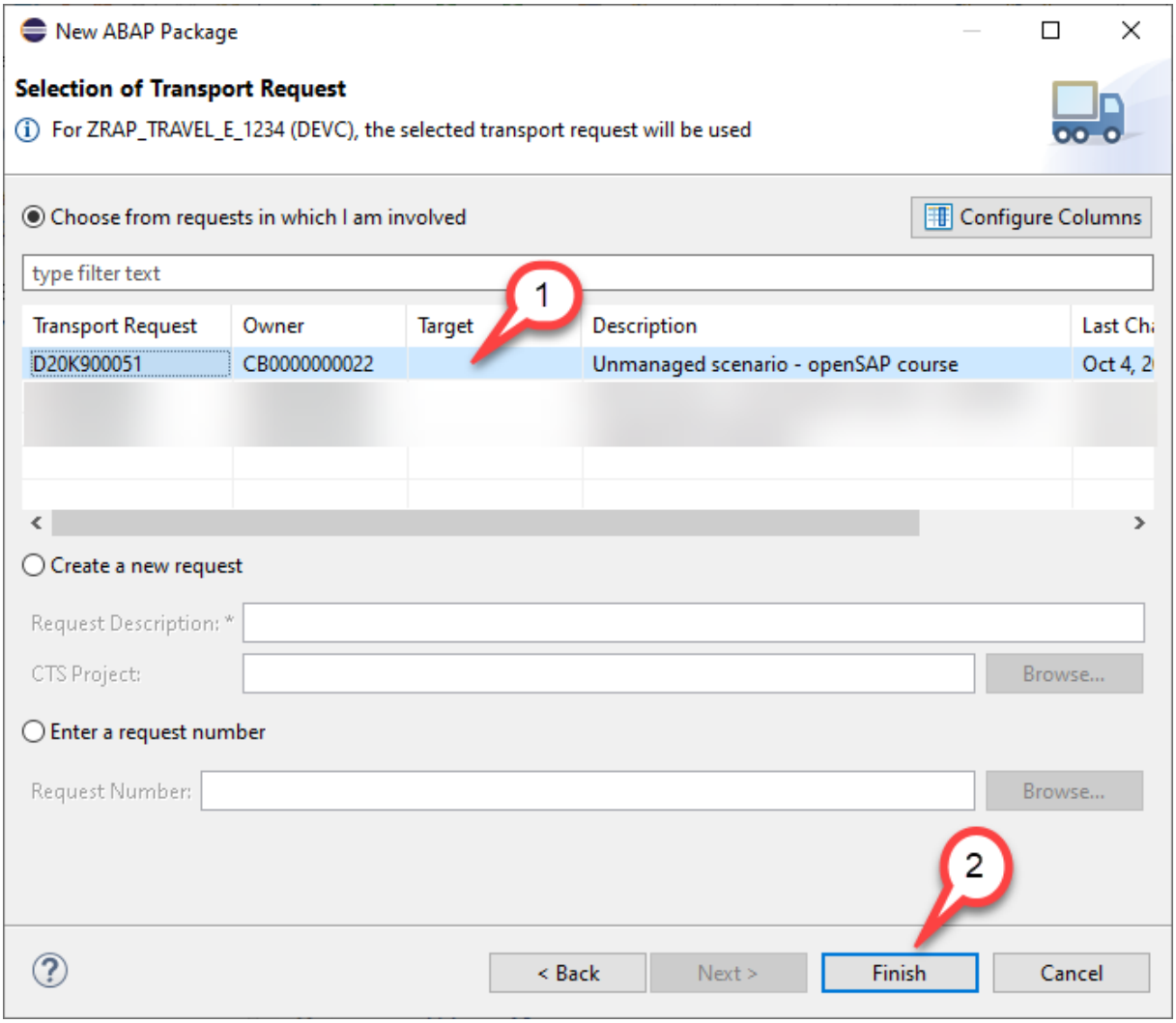
1. Accédez à l'explorateur de projets, faites un clic droit sur votre package **ZRAP_####** et choisissez l'entrée de menu contextuel ***Nouveau > Package ABAP*** .



2. Conservez **ZRAP_TRAVEL_E_####** le nom (où _E indique qu'il s'agit d'un scénario d'extension) et une description pertinente (par exemple, scénario d'extension - Application de rapport de liste de voyages), puis cliquez sur **Suivant** pour continuer.
   Les champs Projet et Superpackage sont automatiquement attribués.



3. Sélectionnez une demande de transport existante ou créez-en une nouvelle, puis choisissez **Terminer** pour créer le nouveau package.

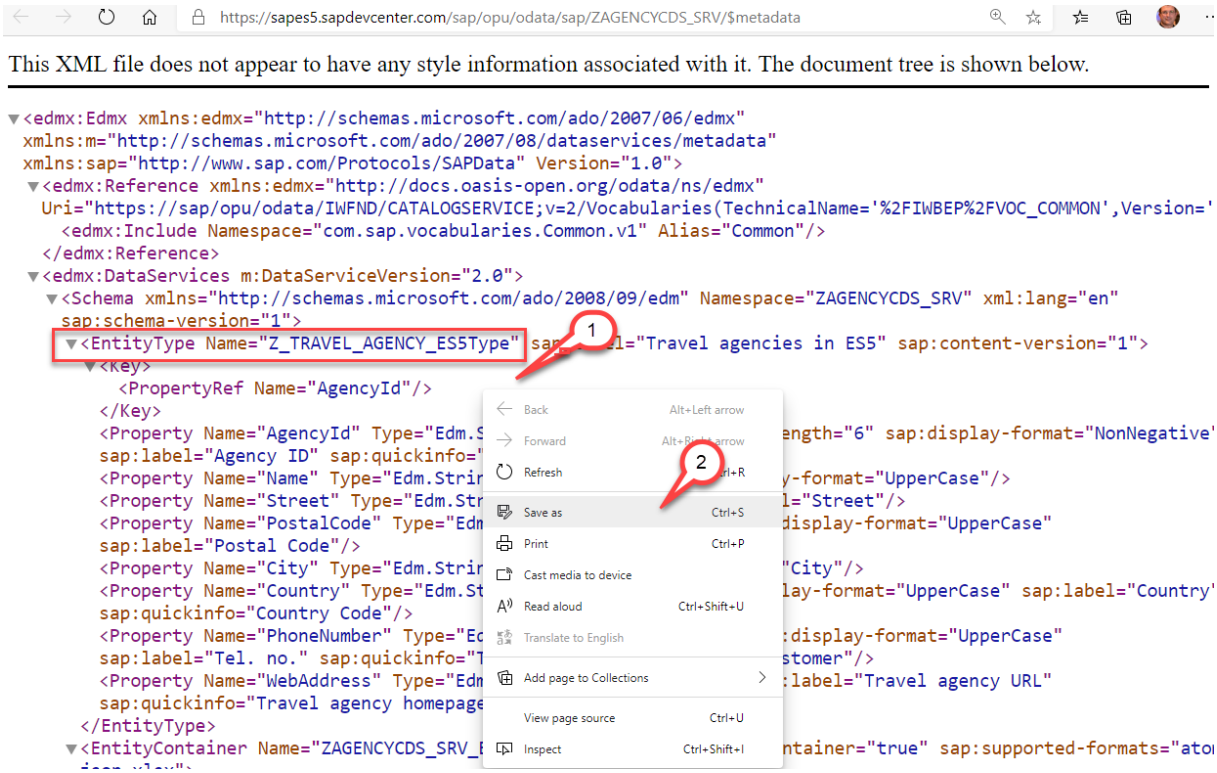


Le package est maintenant créé.

Étape 2 : Téléchargez le fichier $metadata

Le fichier ***$metadata*** du service **OData** que nous souhaitons utiliser doit être téléchargé au format XML. Vous devez donc le télécharger au préalable.

- Cliquez sur l'URL suivante https://sapes5.sapdevcenter.com/sap/opu/odata/sap/ZAGENCYCDS_SRV/$metadata
- Téléchargez le fichier **$metadata** sur votre ordinateur, vous en aurez besoin plus tard dans cet exercice.

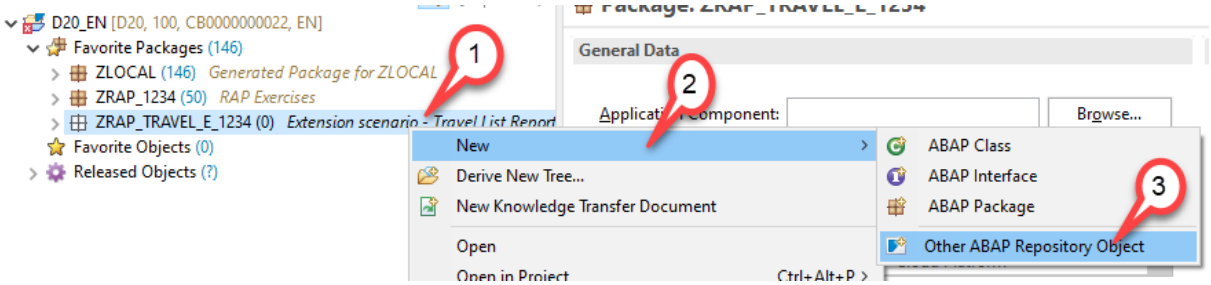


Étape 3 : Créer le modèle de consommation de services

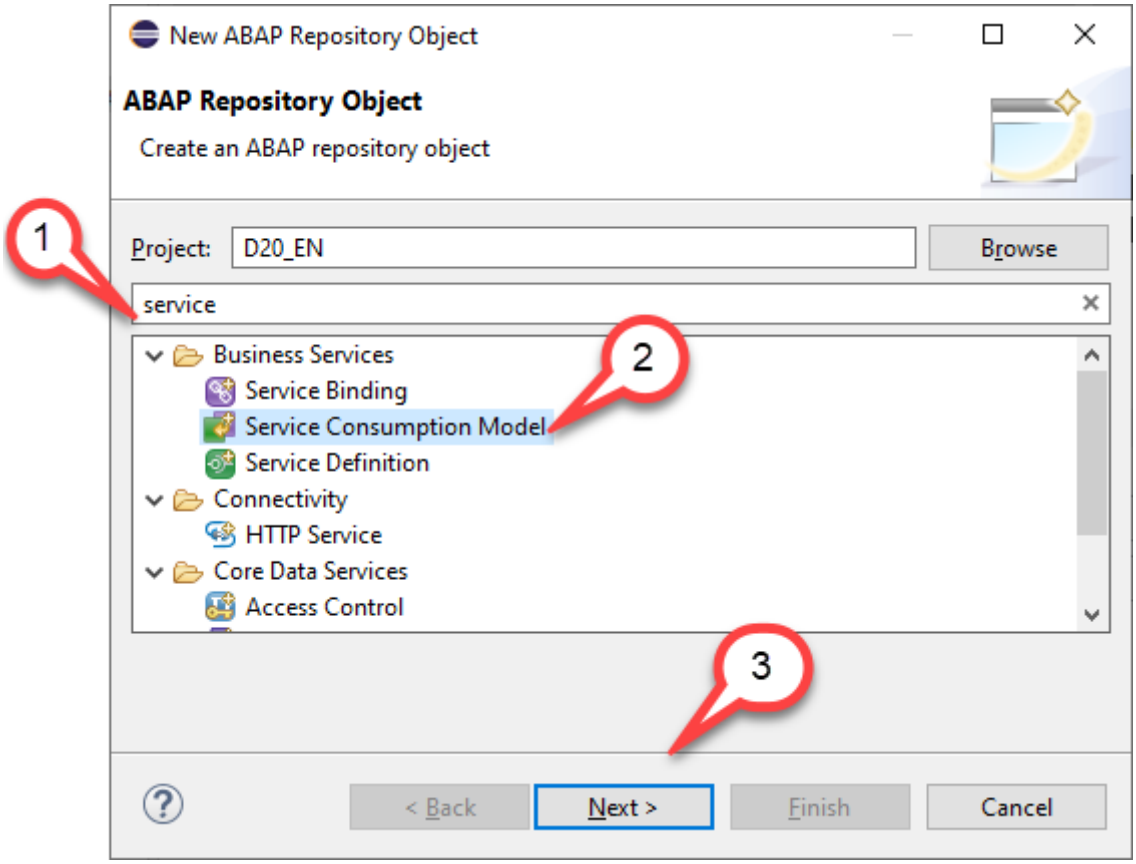
Dans cette étape, nous allons générer un modèle de consommation de services. Ce type d'objet utilise une description d'interface externe en entrée. *OData* et *SOAP* sont actuellement pris en charge. La prochaine version 2011 prévoit également la prise en charge des modèles de consommation de services pour les communications RFC. À partir des informations contenues dans le fichier $metadata ou le fichier WSDL, les objets de référentiel appropriés sont générés (proxy client OData ou proxy SOAP). Grâce à ces objets, vous pourrez écrire du code ABAP permettant de consommer des services OData ou SOAP distants.

Nous commençons par créer un modèle de consommation de services pour un service OData fournissant une liste d'agences de voyages. Ce service réside sur le système public SAP Gateway ES5 et ne nécessite aucune authentification.

1. Passez à ADT et faites un clic droit sur votre package **ZRAP_TRAVEL_E_####**. Sélectionnez **Nouveau > Autre objet de référentiel ABAP**.



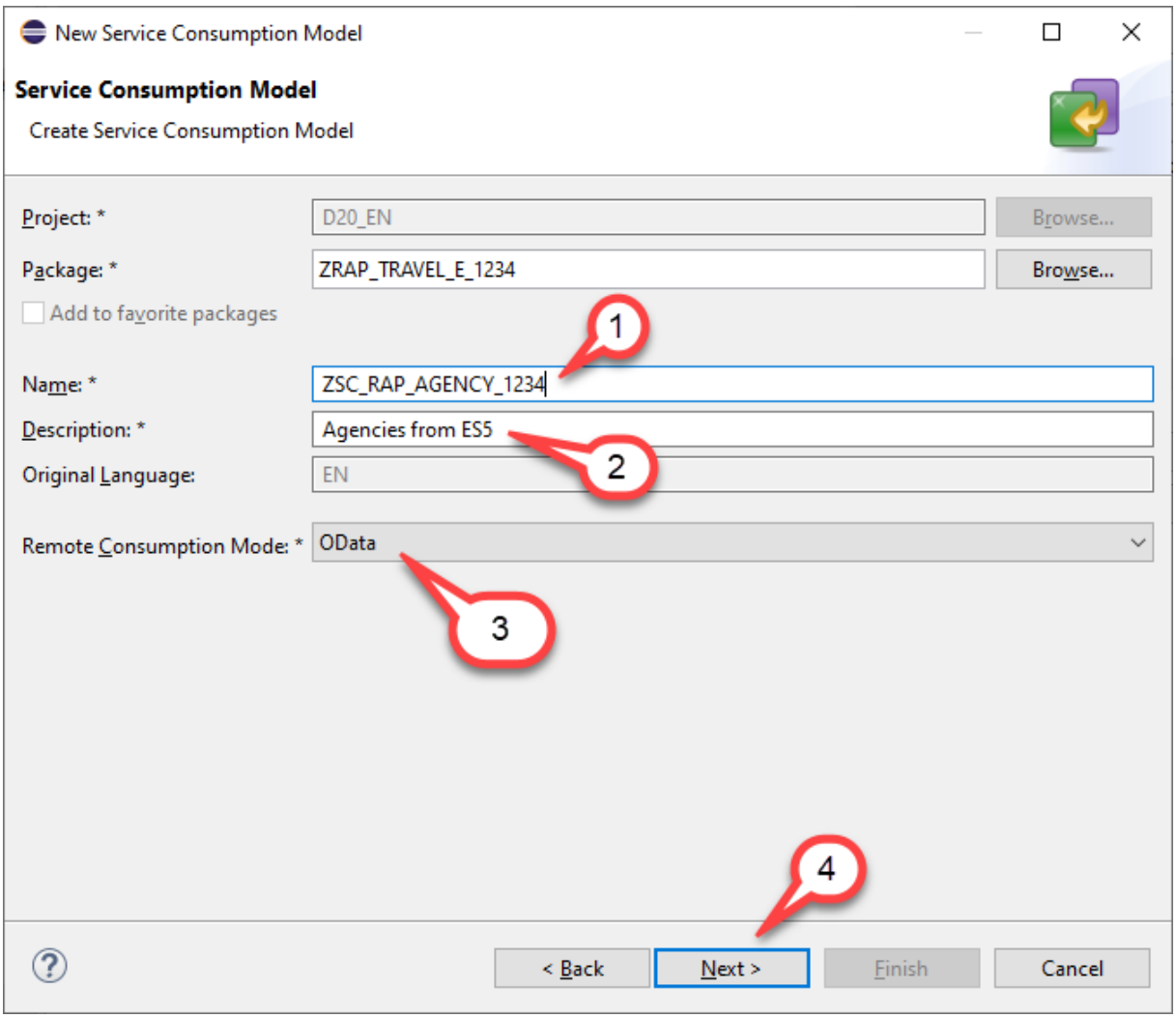
2. Dans la boîte de dialogue Nouvel objet de référentiel ABAP, procédez comme suit
   - Commencez à taper **Service**
   - Dans la liste des objets, sélectionnez **Modèle de consommation de service**
   - Cliquez sur **Suivant**



3. La boîte de dialogue **« Nouveau modèle de consommation de services »** s'ouvre. Saisissez les données suivantes :
   - Nom : **ZSC_RAP_AGENCY_####**
   - Description : **Agencies from ES5**

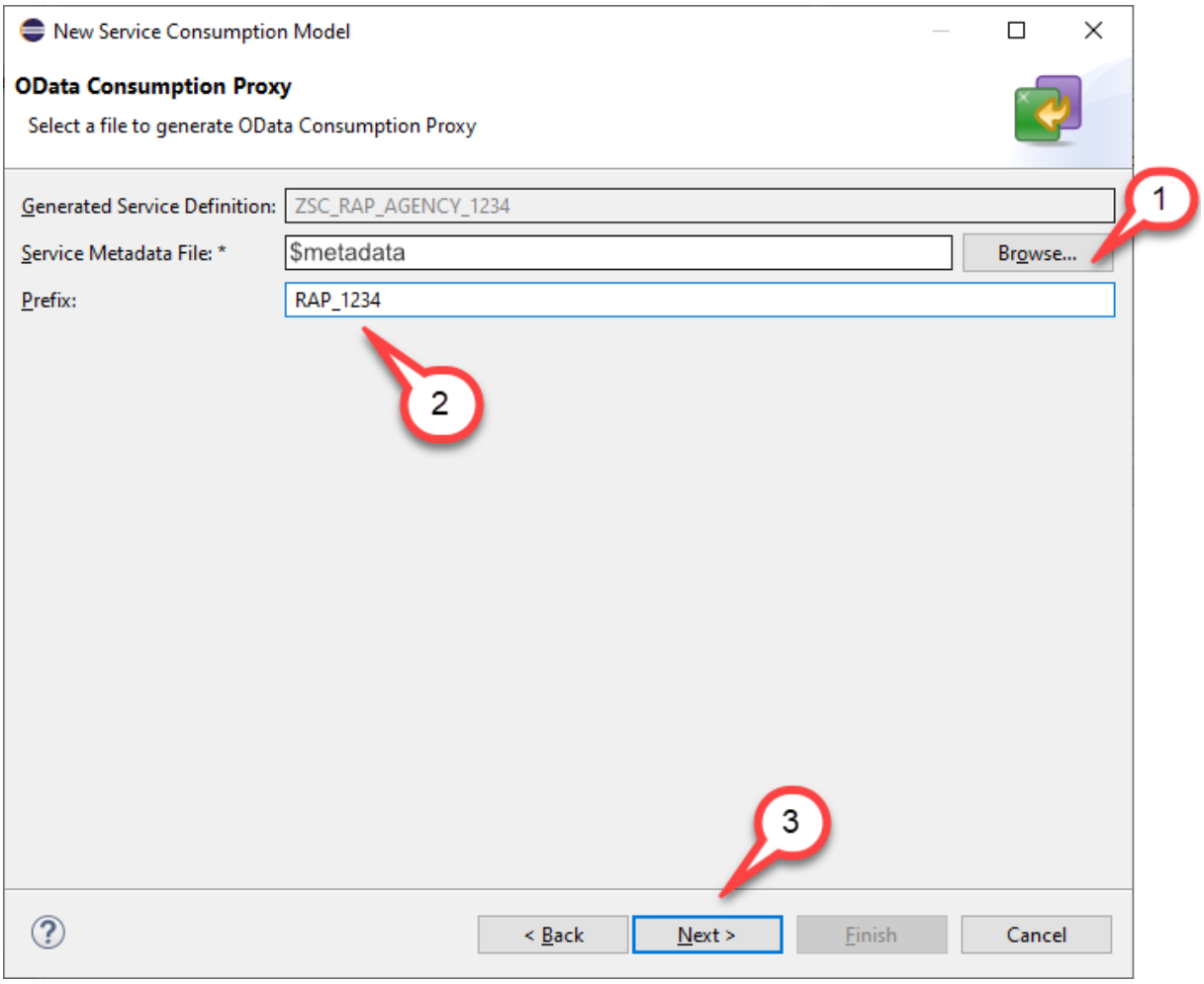
- Modèle de consommation à distance : **OData** (à sélectionner dans la liste déroulante)

Attention :
assurez-vous d'avoir sélectionné **OData** le **mode de consommation à distance** dans la liste déroulante. D'autres options, telles que le service Web et le RFC (prévu), sont également disponibles.



4. Le fichier $metadata du service OData que nous souhaitons utiliser doit être téléchargé au format fichier. Si vous ne l'avez pas encore fait, vous devez le faire maintenant.
   - Cliquez **sur Parcourir** pour sélectionner le fichier $metadata que vous avez téléchargé plus tôt dans cet exercice
   - Préfixe : Entrez **RAP_####**
   - Cliquez sur **Suivant**

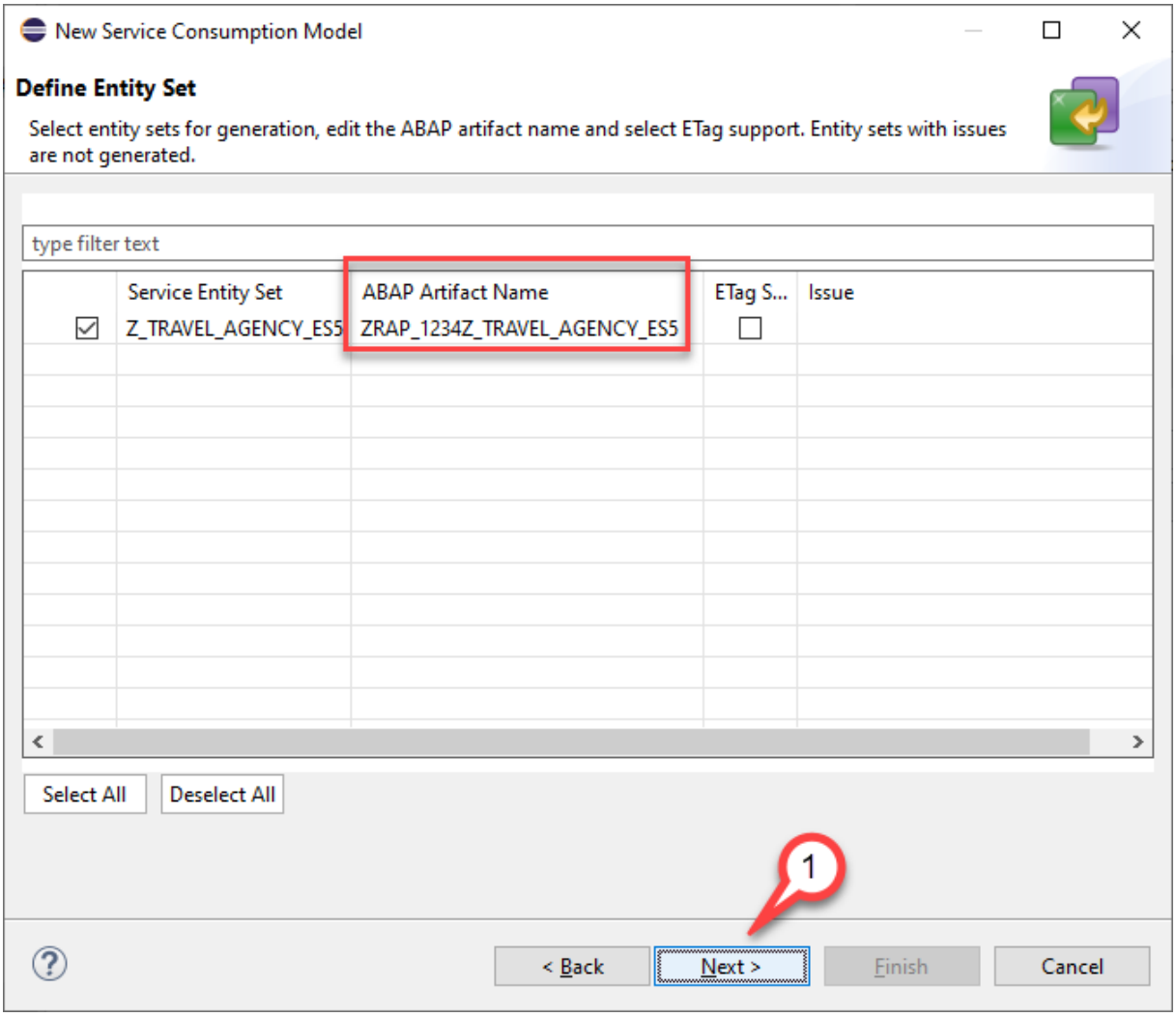
Remarque :
si vous saisissez un préfixe, celui-ci sera ajouté aux noms des entités abstraites générées. Si vous ne sélectionnez pas de préfixe et que l'assistant détecte des conflits de noms, il proposera des noms uniques en ajoutant des caractères arbitraires aux noms des objets du référentiel. Dans tous les cas, vous pourrez modifier les valeurs proposées par l'assistant.



5. Vérifiez le **ABAP Artifact Name**.

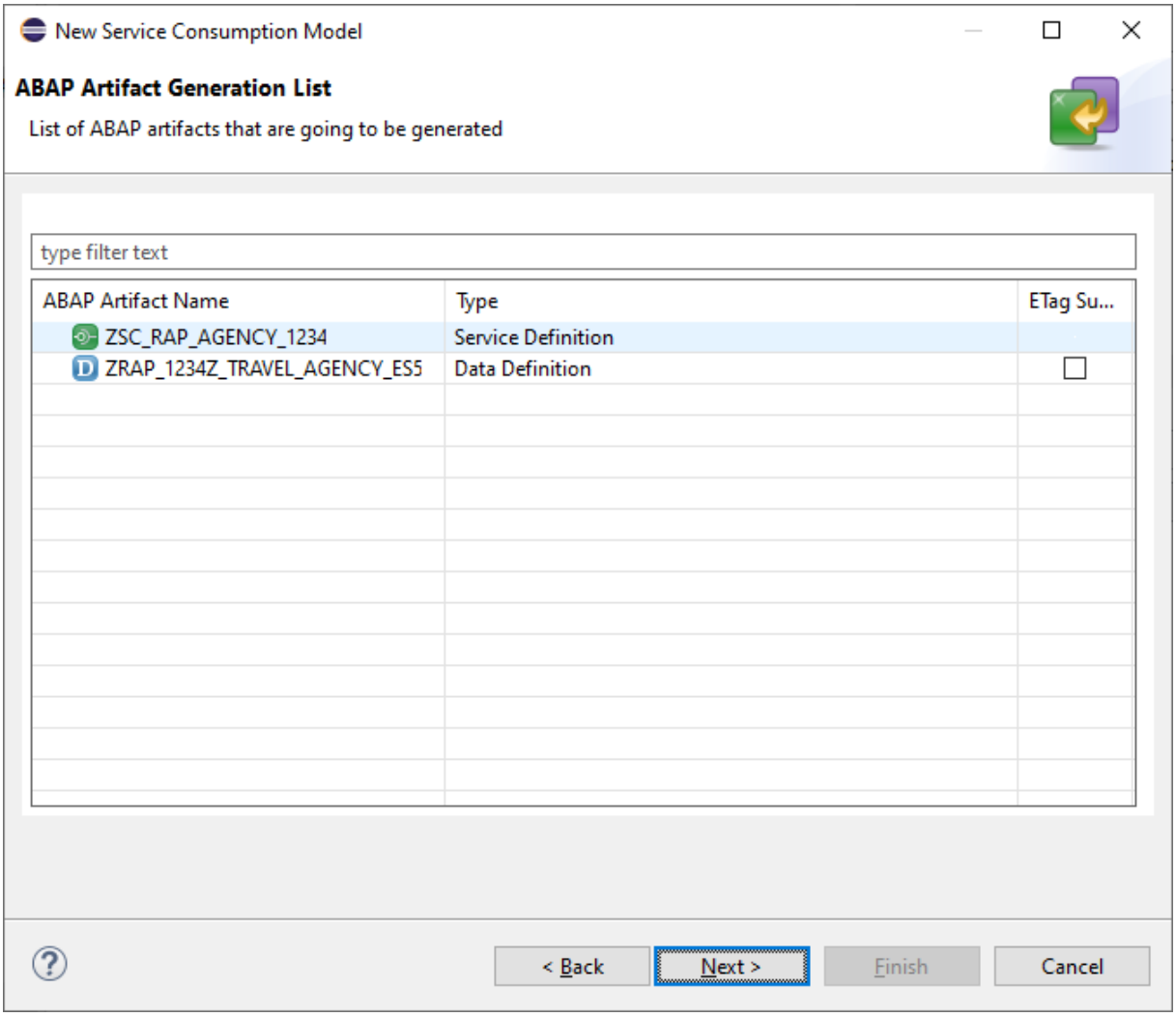
Vous remarquerez que le nom de l'artefact ABAP a été défini sur **ZRAP_####Z_TRAVEL_AGENCY_ES5**puisque nous avons fourni le préfixe **RAP_####**et puisque le nom de l'entité elle-même est **Z_TRAVEL_AGENCY_ES5**.

Appuyez **sur Suivant**.

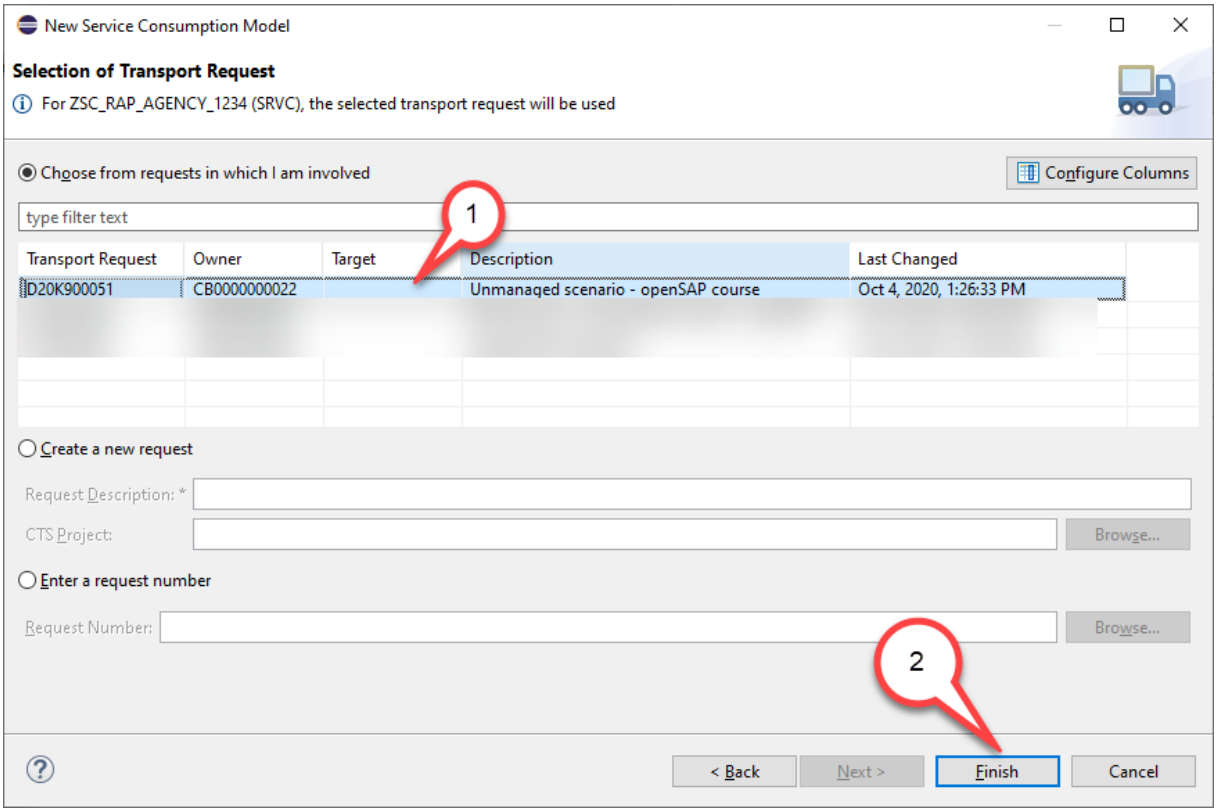


6. L'assistant va maintenant lister les objets du référentiel qui seront générés, à savoir une définition de service et une entité abstraite en plus du modèle de consommation de service.
   - Définition du service : **ZSC_RAP_AGENCY_####**
   - Entité abstraite : **ZRAP_####Z_TRAVEL_AGENCY_ES5**

Cliquez **sur Suivant**.

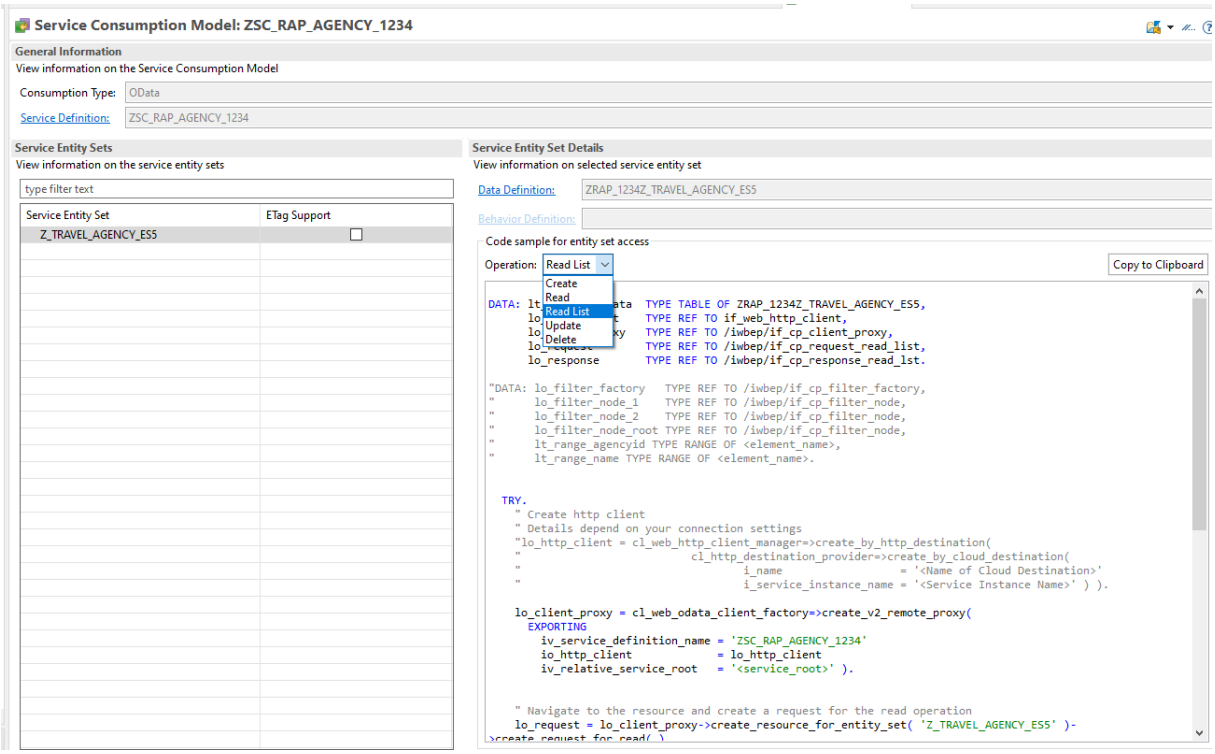


7. Sélection de la demande de transport
    - Sélectionnez ou créez une demande de transport
    - Appuyez sur **Terminer**



8. Examinons brièvement le modèle de consommation de services.

Pour chaque opération (Lire **la liste** , **Lire** , **Créer** , **Mettre à jour** et **Supprimer**), un exemple de code a été créé. Vous pouvez l'utiliser pour appeler le service OData. Pour récupérer une liste d'agences, sélectionnez l'opération « **Lire la liste »** et cliquez sur le bouton **« Copier dans le presse-papiers »** . Ce code sera utilisé à l'étape suivante, où nous créerons une application console pour tester l'appel au service OData distant.

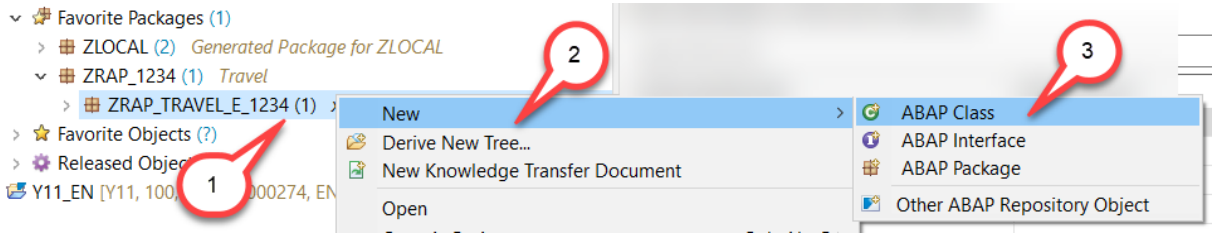


Étape 4 : Créer une application console pour tester le service OData

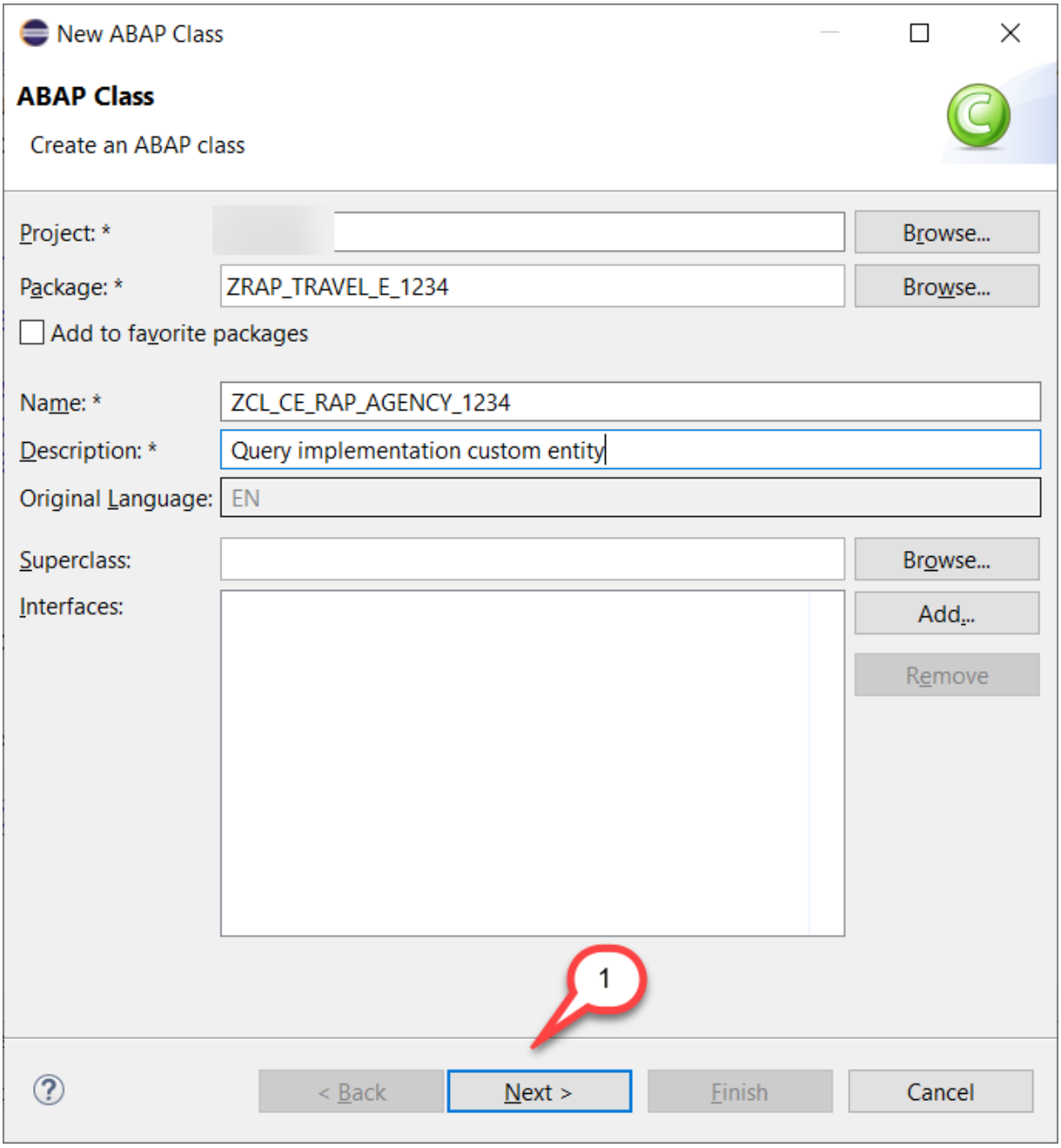
Nous pouvons maintenant tester le modèle de consommation de services en créant une petite application console **ZCL_CE_RAP_AGENCY_####** qui implémente l'interface **if_oo_adt_classrun** . Cette étape supplémentaire est utile, car elle permet de vérifier plus facilement le fonctionnement de la consommation OData et de déboguer une application console bien plus facilement que de tester le code dans l'objet métier RAP complet.

Veuillez noter que
nous utiliserons cette classe à un stade ultérieur également comme implémentation pour notre requête personnalisée et nous choisissons donc un nom **ZCL_CE_RAP_AGENCY_####** qui contient déjà le nom de l'entité personnalisée à créer. **CE** indique que cette classe agira comme une implémentation de requête pour une *entité personnalisée*.

1. Faites un clic droit sur le dossier **Bibliothèque de code source** et sélectionnez **Nouveau > Classe ABAP**.

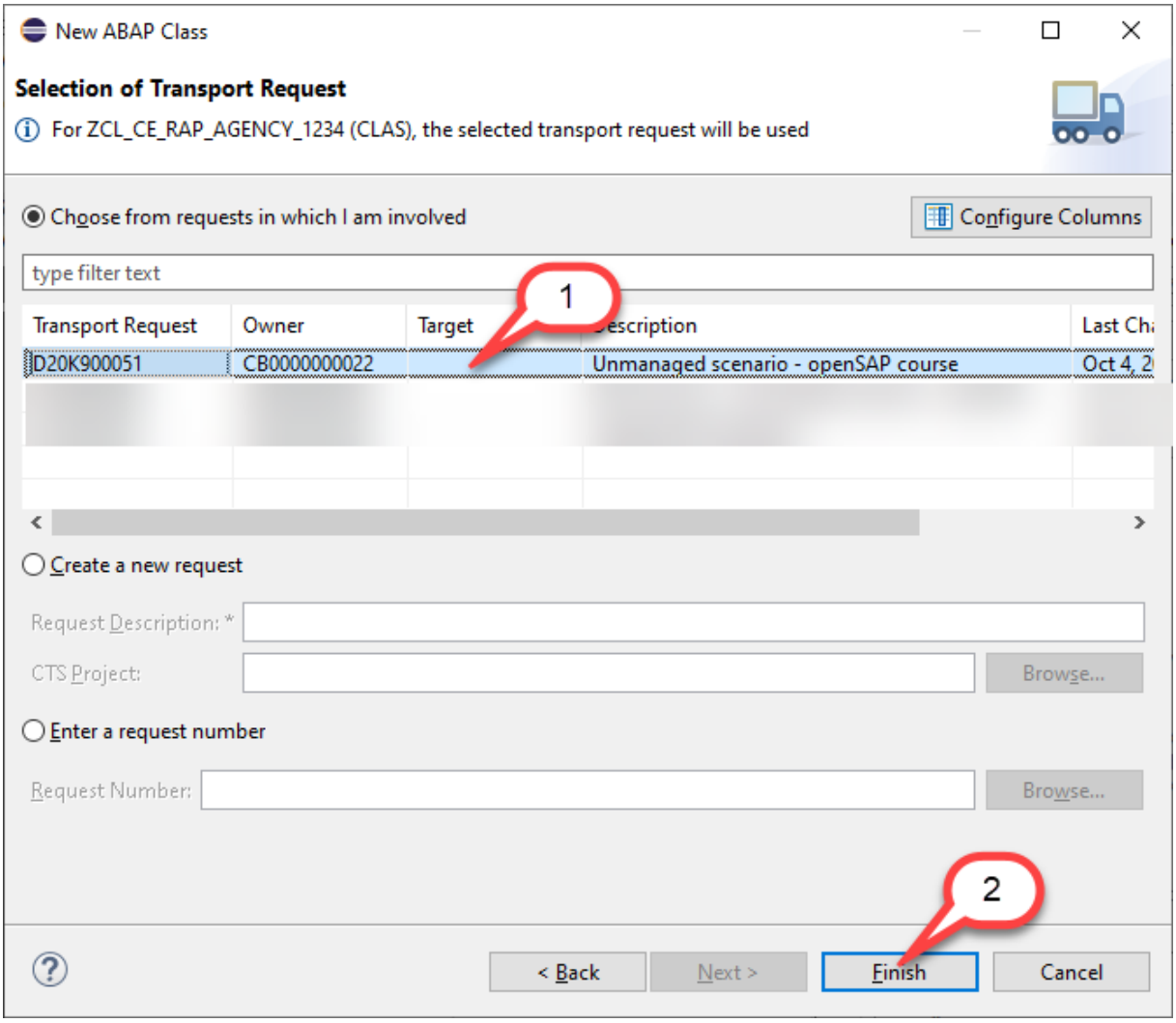


2. La boîte de dialogue **« Nouvelle classe ABAP »** s'ouvre. Saisissez les informations suivantes :
   - Nom : **ZCL_CE_RAP_AGENCY_####**
   - Description : **Query implementation custom entity**
   - Cliquez **sur Ajouter**



3. Vérifiez l'entrée et appuyez sur **Suivant**

4. Sélection de la demande de transport
   - Sélectionnez ou créez une demande de transport
   - Cliquez sur **Terminer**



5. Remplacez le code par l'exemple de code fourni ci-dessous

ZCL_CE_RAP_AGENCY_1234

```
1  CLASS zcl_ce_rap_agency_1234 DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5
6    PUBLIC SECTION.
7    PROTECTED SECTION.
8    PRIVATE SECTION.
9  ENDCLASS.
10
11
12
13 CLASS zcl_ce_rap_agency_1234 IMPLEMENTATION.
14
15 ENDCLASS.
```

Replace code with the code provided in the script

```
CLASS zcl_ce_rap_agency_#### DEFINITION

  PUBLIC

  FINAL

  CREATE PUBLIC .


  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun.


    TYPES t_agency_range TYPE RANGE OF zrap_####z_travel_agency_es5-agencyid.

    TYPES t_business_data TYPE TABLE OF zrap_####z_travel_agency_es5.


    METHODS get_agencies

      IMPORTING
```

```
        filter_cond        TYPE if_rap_query_filter=>tt_name_range_pairs
OPTIONAL

        top                TYPE i OPTIONAL

        skip               TYPE i OPTIONAL

        is_data_requested  TYPE abap_bool

        is_count_requested TYPE abap_bool

      EXPORTING

        business_data      TYPE t_business_data

        count              TYPE int8

      RAISING

        /iwbep/cx_cp_remote

        /iwbep/cx_gateway

        cx_web_http_client_error

        cx_http_dest_provider_error

      .


  PROTECTED SECTION.

  PRIVATE SECTION.

ENDCLASS.





CLASS zcl_ce_rap_agency_#### IMPLEMENTATION.


  METHOD if_oo_adt_classrun~main.


    DATA business_data TYPE t_business_data.
```

```
    DATA count TYPE int8.

    DATA filter_conditions  TYPE if_rap_query_filter=>tt_name_range_pairs .

    DATA ranges_table TYPE if_rap_query_filter=>tt_range_option .

    ranges_table = VALUE #( (  sign = 'I' option = 'GE' low = '070015' ) ).

    filter_conditions = VALUE #( ( name = 'AGENCYID'  range = ranges_table ) ).


    TRY.

        get_agencies(

          EXPORTING

            filter_cond        = filter_conditions

            top                = 3

            skip               = 1

            is_count_requested = abap_true

            is_data_requested  = abap_true

          IMPORTING

            business_data  = business_data

            count          = count

          ) .

        out->write( |Total number of records = { count }| ) .

        out->write( business_data ).

      CATCH cx_root INTO DATA(exception).

        out->write( cl_message_helper=>get_latest_t100_exception( exception )-
>if_message~get_longtext( ) ).

    ENDTRY.




  ENDMETHOD.
```

```
  METHOD get_agencies.


    DATA: filter_factory   TYPE REF TO /iwbep/if_cp_filter_factory,

          filter_node      TYPE REF TO /iwbep/if_cp_filter_node,

          root_filter_node TYPE REF TO /iwbep/if_cp_filter_node.


    DATA: http_client        TYPE REF TO if_web_http_client,

          odata_client_proxy TYPE REF TO /iwbep/if_cp_client_proxy,

          read_list_request  TYPE REF TO /iwbep/if_cp_request_read_list,

          read_list_response TYPE REF TO /iwbep/if_cp_response_read_lst.


    DATA service_consumption_name TYPE
cl_web_odata_client_factory=>ty_service_definition_name.


    DATA(http_destination) = cl_http_destination_provider=>create_by_url( i_url =
'https://sapes5.sapdevcenter.com' ).

    http_client = cl_web_http_client_manager=>create_by_http_destination(
i_destination = http_destination ).


    service_consumption_name = to_upper( 'ZSC_RAP_AGENCY_####' ).


    odata_client_proxy = cl_web_odata_client_factory=>create_v2_remote_proxy(

      EXPORTING

        iv_service_definition_name = service_consumption_name

        io_http_client             = http_client

        iv_relative_service_root   = '/sap/opu/odata/sap/ZAGENCYCDS_SRV/' ).
```

```
    " Navigate to the resource and create a request for the read operation

    read_list_request = odata_client_proxy->create_resource_for_entity_set(
'Z_TRAVEL_AGENCY_ES5' )->create_request_for_read( ).



    " Create the filter tree

    filter_factory = read_list_request->create_filter_factory( ).

    LOOP AT  filter_cond  INTO DATA(filter_condition).

      filter_node  = filter_factory->create_by_range( iv_property_path     =
filter_condition-name

                                                                  it_range     =
filter_condition-range ).

      IF root_filter_node IS INITIAL.

        root_filter_node = filter_node.

      ELSE.

        root_filter_node = root_filter_node->and( filter_node ).

      ENDIF.

    ENDLOOP.


    IF root_filter_node IS NOT INITIAL.

      read_list_request->set_filter( root_filter_node ).

    ENDIF.


    IF is_data_requested = abap_true.

      read_list_request->set_skip( skip ).

      IF top > 0 .

        read_list_request->set_top( top ).

      ENDIF.

    ENDIF.
```

```
    IF is_count_requested = abap_true.

      read_list_request->request_count(  ).

    ENDIF.


    IF is_data_requested = abap_false.

      read_list_request->request_no_business_data(  ).

    ENDIF.


    " Execute the request and retrieve the business data and count if requested

    read_list_response = read_list_request->execute( ).

    IF is_data_requested = abap_true.

      read_list_response->get_business_data( IMPORTING et_business_data =
business_data ).

    ENDIF.

    IF is_count_requested = abap_true.

      count = read_list_response->get_count(  ).

    ENDIF.


  ENDMETHOD.




ENDCLASS.
```

6. **Le code expliqué**

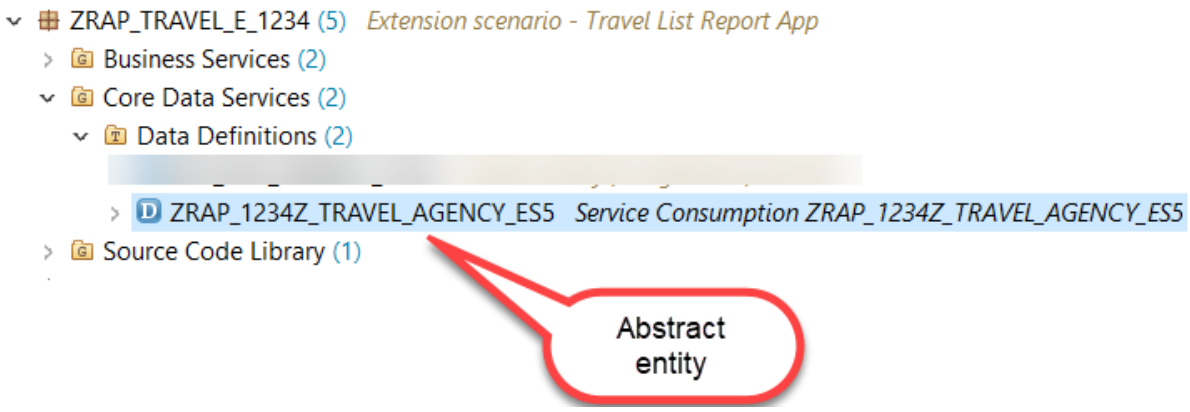
Jetons un œil à l'implémentation de notre classe de test.

Dans la section publique, nous trouvons deux définitions **de TYPES**.

Le type t_agency_range permet de fournir des conditions de filtrage pour les agences sous forme d'options SELECT à la méthode **get_agencies()** . Le deuxième type **t_business_data** permet de récupérer les données métiers renvoyés par notre service OData distant.

Les deux types sont basés sur l'entité abstraite **ZRAP_####Z_TRAVEL_AGENCY_ES5** qui a été générée lorsque vous avez généré le modèle de consommation de services.

Remarque :
si vous n'avez pas utilisé le préfixe, RAP_####le nom de l'entité abstraite peut être différent. Dans ce cas, remplacez toutes les occurrences de **ZRAP_####Z_TRAVEL_AGENCY_ES5** par le nom de l'entité abstraite générée dans votre package * **ZRAP_TRAVEL_E_####** .



La méthode **get_agencies()** accepte les conditions de filtrage sous forme de SELECT-OPTIONS via le paramètre d'importation **it_filter_cond** . De plus, il est possible de fournir des valeurs pour **top** et **skip** afin d'optimiser la pagination côté client.

En utilisant les paramètres **is_data_requested** et **is_count_requested,** nous indiquons au proxy client OData si les données doivent être demandées au backend ou non et nous pouvons spécifier si, en plus des données commerciales, le nombre total d'enregistrements de cette entité doit également être demandé.

La méthode publique **get_agencies( )** permet de récupérer les données du service OData distant. Comme il n'est pas possible d'exploiter le service de destination dans les systèmes d'essai, nous utiliserons la méthode **cl_http_destination_provider=>create_by_url** , qui permet de créer un objet de destination http basé sur l'URL cible. Comme URL cible, nous choisissons l'URL racine https://sapes5.sapdevcenter.com du système ES5, car l'URL relative pointant vers le service OData sera ajoutée lors de la création du proxy client OData.

Dans la méthode **get_agencies(),** nous créons un objet proxy client OData odata_client_proxy, qui crée un objet de requête read_list_requestpour l'opération de lecture. Cet read_list_requestobjet récupère tous les paramètres, tels que le filtre, ainsi que les informations concernant les données métier et le nombre total d'entrées demandées au client. Il read_list_requestrenvoie le nombre total

d'entrées via la variable countet les données métier via la table interne business_data. Si l'appel réussit, le nombre total d'entrées et les données métier sont écrits dans la console. En cas d'exception, le message d'erreur est récupéré et écrit dans la console.

Remarque :
dans un environnement ABAP SAP Cloud Platform non-version d'essai, il est possible d'exploiter le service de destination de l'environnement Cloud Foundry sous-jacent et d'utiliser l'instruction **cl_http_destination_provider=>create_by_cloud_destination** pour générer une destination http dans l'environnement ABAP en fonction de ces paramètres.
Comme il n'est pas possible d'exploiter le service de destination dans les systèmes d'essai, nous utiliserons la méthode **create_by_http_destination** , qui permet de créer un objet client http basé sur l'URL cible.
Nous utilisons ici l'URL racine https://sapes5.sapdevcenter.com du système ES5, car l'URL relative sera ajoutée lors de la création du proxy client OData.

7. Vous pouvez maintenant exécuter l'application console en appuyant sur **F9**.

```
ZCL_CE_RAP_AGENCY_1234 ▸ GET_AGENCIES
1  CLASS zcl_ce_rap_agency_1234 DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5
6    PUBLIC SECTION.
7      INTERFACES if_oo_adt_classrun.
8
9      TYPES t_agency_range TYPE RANGE OF zrap_1234z_travel_agency_es5-agencyid.
10     TYPES t_business_data TYPE TABLE OF zrap_1234z_travel_agency_es5.
11
12     METHODS get_agencies
13       IMPORTING
14         filter_cond        TYPE if_rap_query_filter=>tt_name_range_pairs   OPTIONAL
15         top                TYPE i OPTIONAL
16         skip               TYPE i OPTIONAL
17         is_data_requested  TYPE abap_bool
18         is_count_requested TYPE abap_bool
19       EXPORTING
20         business_data      TYPE t_business_data
21         count              TYPE int8
22       RAISING
23         /iwbep/cx_cp_remote
24         /iwbep/cx_gateway
```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Console

ABAP Console

```
Total number of records = 50
Table
AGENCYID  NAME                 NAME_VC  STREET             STREET_VC  POSTALCODE  POSTALCODE_VC  CITY
099902    ES5   Fly High                Berliner Allee 11             40880                      Duesseldorf
099903    ES5   Happy Hopping           Calvinstr. 36                 13467                      Berlin
099904    ES5   Pink Panther            Auf der Schanz 54             65936                      Frankfurt
```

Résumé

Dans cette unité, vous avez appris à créer le modèle de consommation de services pour accéder aux données d'un système externe via OData.

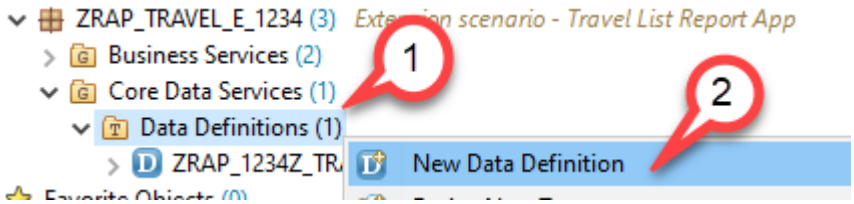
6.2 Définition de l'entité CDS personnalisée (Unité 2)

Objectif de l'unité

Créer une entité CDS personnalisée servant de wrapper pour consommer des données issues d'un service OData distant, en préparation à son implémentation via une classe RAP dans l'unité suivante.

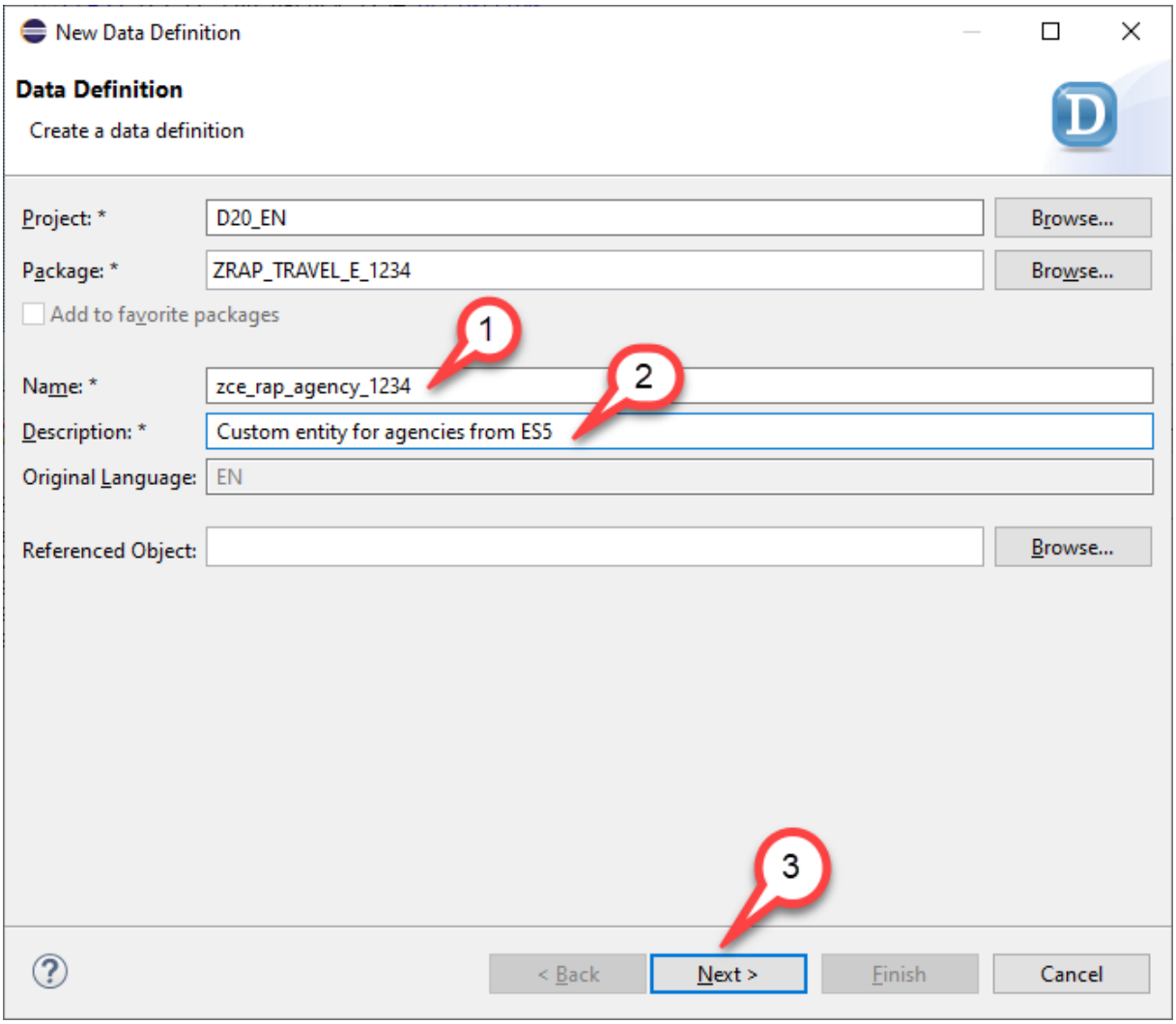
Créer une entité personnalisée

1. Cliquez avec le bouton droit sur le dossier **Définition de données** et sélectionnez **Nouvelle définition de données.**



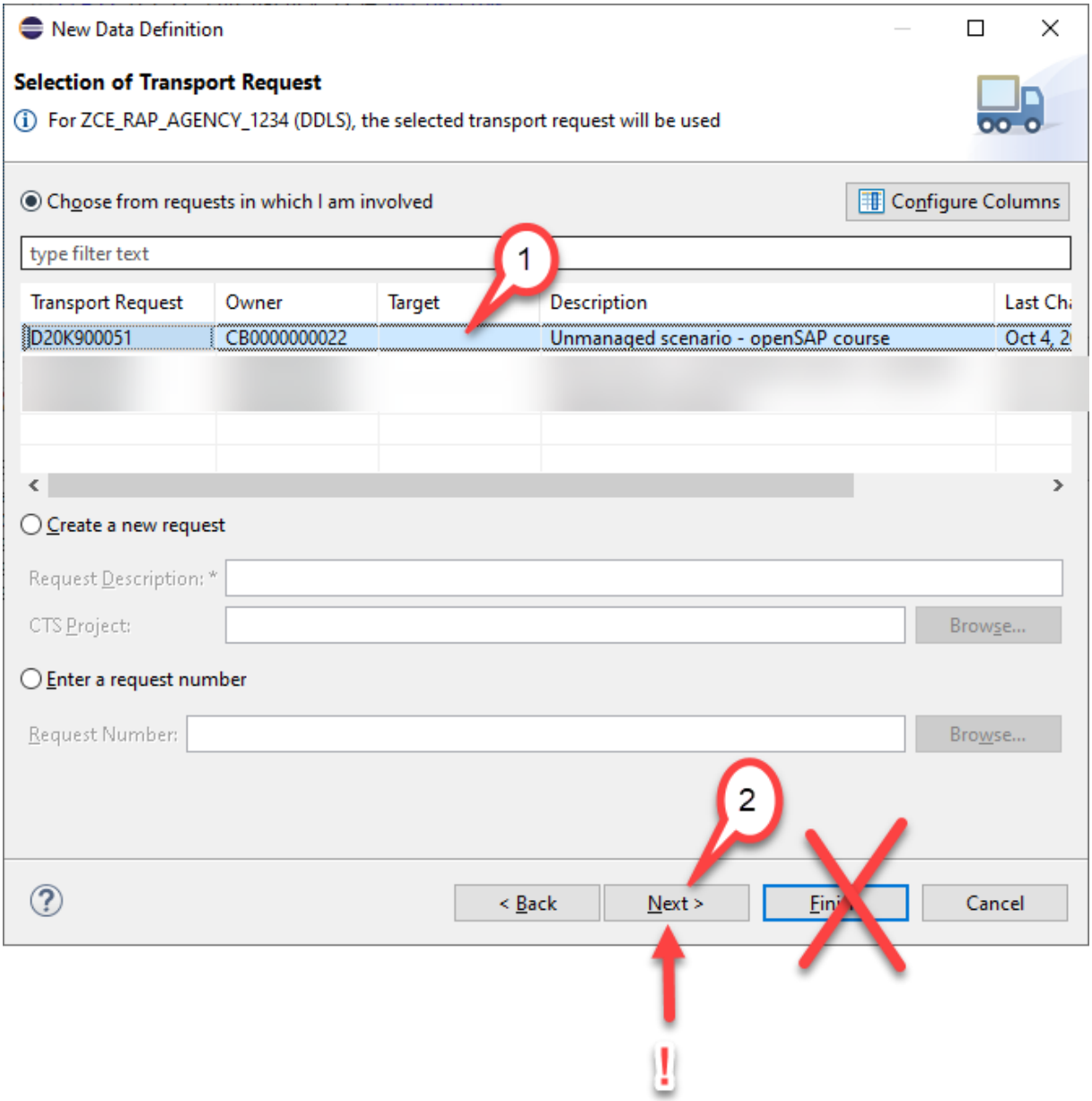
1. Commençons par créer une nouvelle définition de données ZCE_RAP_AGENCY_####à l'aide du modèle pour une **entité personnalisée**.
2. La boîte de dialogue **Nouvelle définition des données** s'ouvre
   - Nom : ZCE_RAP_AGENCY_####
   - Description : Custom entity for agencies from ES5

Appuyez **sur Suivant**



3. Sélection d'une demande de transport
   - Sélectionnez ou créez une demande de transport
   - **!!! SEULEMENT !!!** Appuyez sur *Suivant*. N'appuyez **PAS** sur *Terminer*.

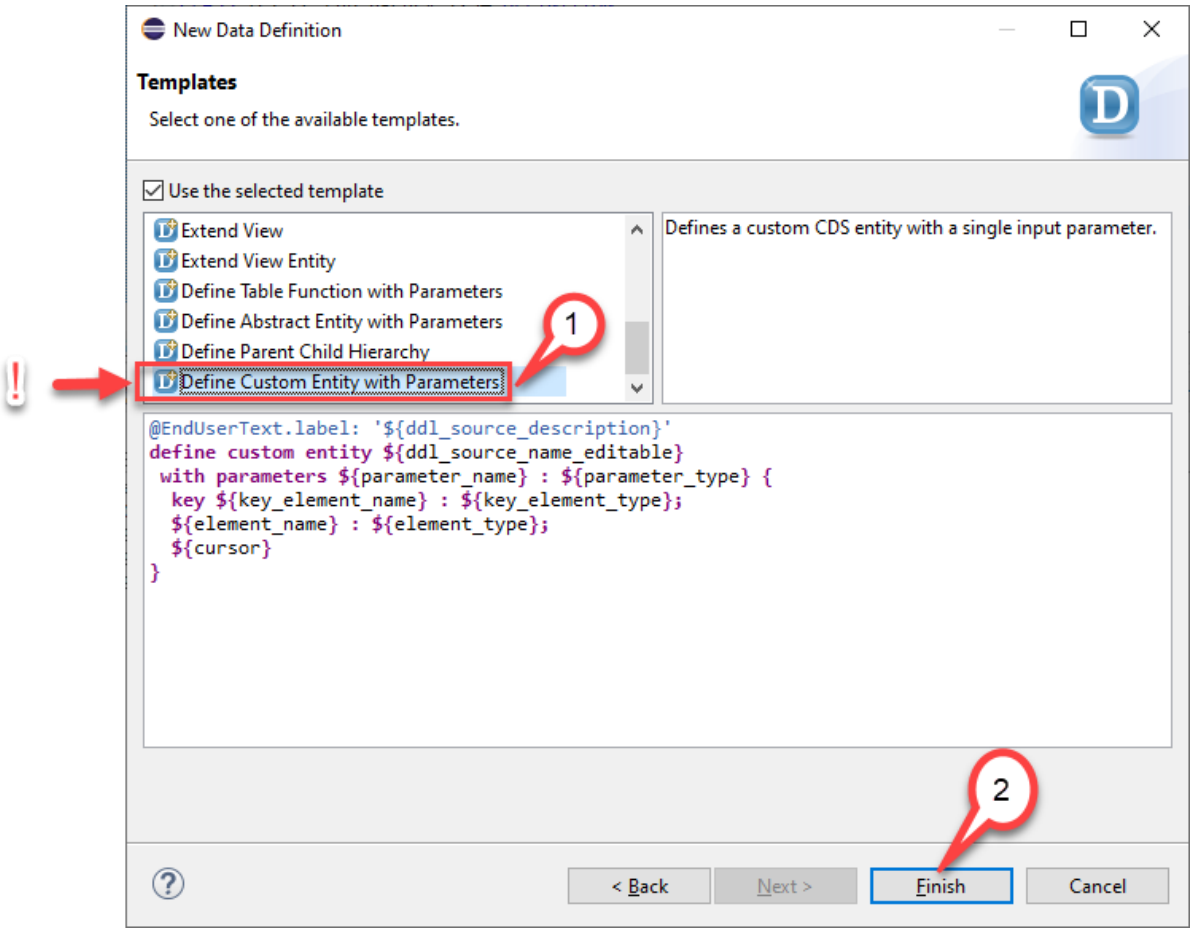
Attention :
si vous cliquez sur **Terminer** au lieu de **Suivant**, l'assistant utilisera le modèle utilisé lors de sa dernière utilisation par le développeur.
Pour être sûr du modèle sélectionné, il **est impératif** de cliquer sur **Suivant** et non **sur Terminer,** ce qui ignorerait l'étape de sélection du modèle.



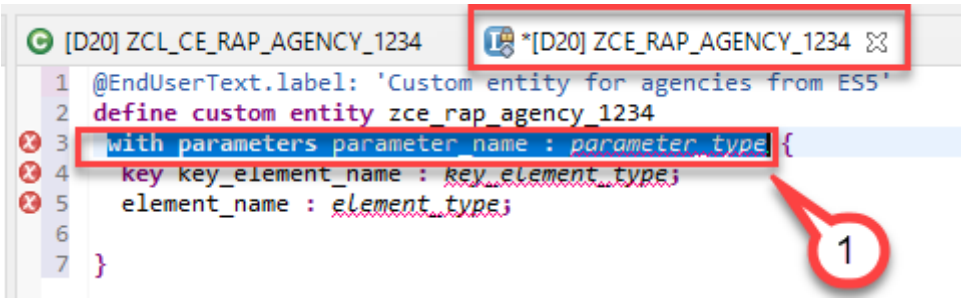
4. Sélectionner un modèle
   - Utilisez la barre de défilement pour naviguer dans la liste
   - Sélectionnez le modèleDefine Custom Entity with Parameters
   - Appuyez sur **Terminer**

Attention :
il n'existe qu'un modèle pour une entité personnalisée avec paramètres. Cela n'a aucune importance.

Nous utilisons ce modèle et supprimons l'instruction *avec les paramètres parameter_name : parameter_type* .

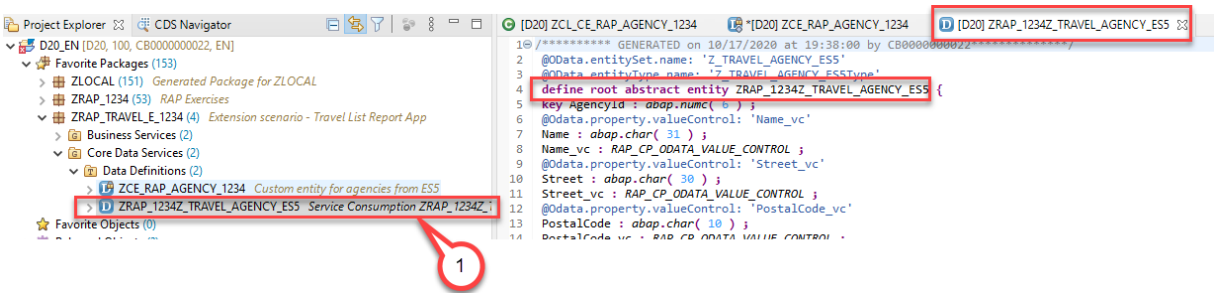


5. Modifier le code source de l'entité **personnalisée** ZCE_RAP_AGENCY_####
   - Supprimer la déclaration avec les paramètres parameter_name : parameter_type

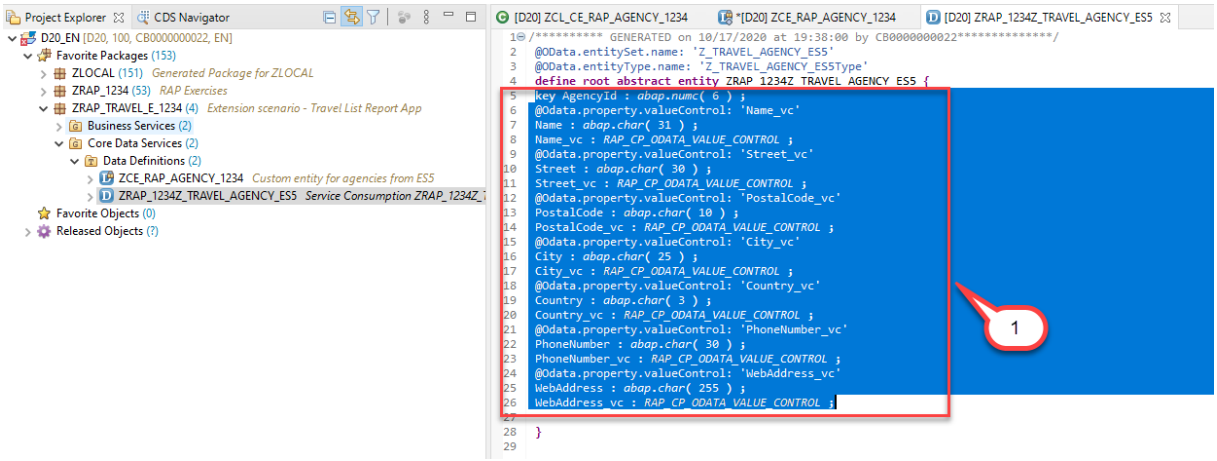


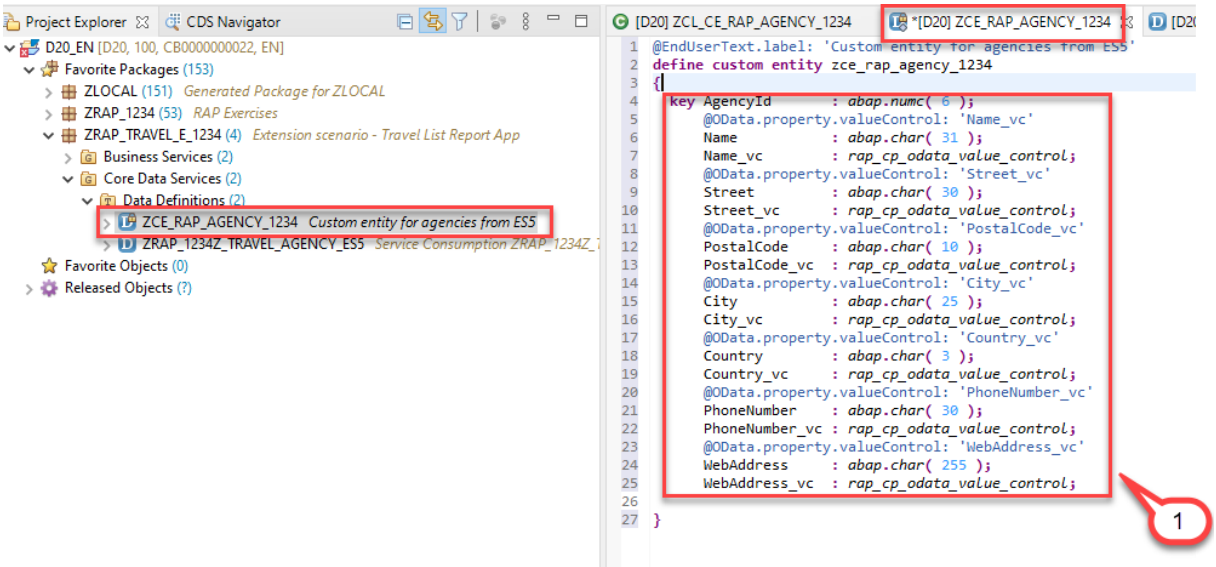
6. Maintenant, ouvrez l'entité **abstraite** que vous avez créée dans l'unité 2 dans l'éditeur de code source

- Accédez à l'entité **abstraite** ZRAP_####Z_TRAVEL_AGENCY_ES5 dans l' *explorateur de projets*
- Double-cliquez dessus pour ouvrir l'éditeur de code source



7. Copier la liste des champs de l'entité abstraite vers l'entité personnalisée
   - Copiez la liste des champs de l'entité abstraite et collez-la entre les accolades comme nouvelle liste de champs de l'entité personnalisée, remplaçant ainsi celle fournie par le modèle de code. Vous pouvez utiliser le formateur de code source (**MAJ+F1**) pour formater le code source.

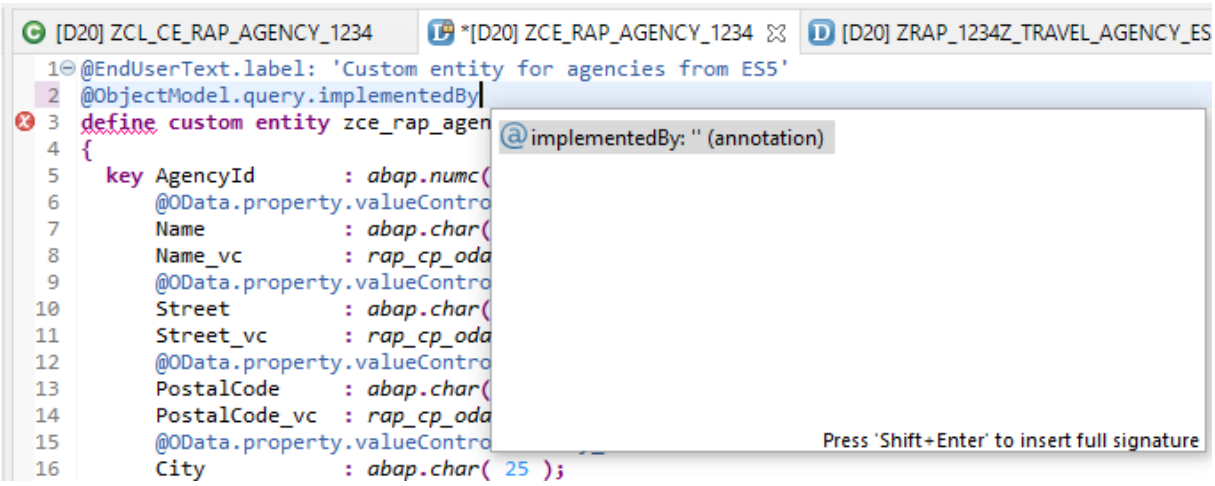




8. Ajoutez l'annotation @ObjectModel.Query.ImplementedBy au code source DDL

```
@ObjectModel.query.implementedBy : 'ABAP:ZCL_CE_RAP_AGENCY_####'
```

De cette façon, nous réutiliserons la classe que nous avons créée dans l'unité 2.



9. Activez vos modifications

[X11] ZCE_RAP_AGENCY_1234

```
1 @EndUserText.label: 'Custom entity for agencies from ES5'
2 @ObjectModel.query.implementedBy: 'ABAP:ZCL_CE_RAP_AGENCY_1234'
3 define custom entity ZCE_RAP_AGENCY_1234
4 {
5   key AgencyId        : abap.numc( 6 );
6       @OData.property.valueControl: 'Name_vc'
7       Name            : abap.char( 31 );
8       Name_vc         : rap_cp_odata_value_control;
9       @OData.property.valueControl: 'Street_vc'
10      Street          : abap.char( 30 );
11      Street_vc       : rap_cp_odata_value_control;
12      @OData.property.valueControl: 'PostalCode_vc'
13      PostalCode      : abap.char( 10 );
14      PostalCode_vc   : rap_cp_odata_value_control;
```

Résumé

Dans cette unité, vous avez créé une entité personnalisée basée sur l'entité abstraite qui a été générée lors de l'activation du modèle de consommation de services.

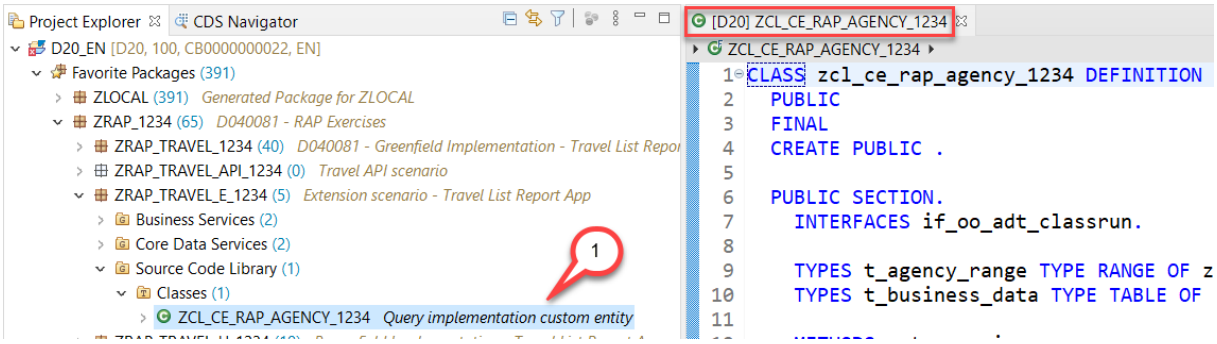
6.3 Création de la classe d'implémentation (Unité 3)

Objectif de l'unité

Créer une classe d'implémentation pour une entité CDS personnalisée, en ajoutant l'interface IF_RAP_QUERY_PROVIDER afin de fournir les données dynamiquement depuis un service OData distant via la méthode SELECT.

Création de l'implémentation

1. Ouvrez la classe **ZCL_CE_RAP_AGENCY_####** dans l'éditeur de code source.
    - Développez votre package **ZRAP_TRAVEL_E_#### > Bibliothèque de code source > Classes** et double-cliquez surZCL_CE_RAP_AGENCY_####



2. Ajouter une interface **IF_RAP_QUERY_PROVIDER**à la classe d'implémentation de requête **ZCL_CE_RAP_AGENCY_####**
    - Ajoutez la déclaration suivante dans la section publique

```
INTERFACES if_rap_query_provider.
```

    - Vous pouvez utiliser la saisie semi-automatique du code **CTRL+Espace** pour sélectionner l'interface **if_rap_query_provider**.

```
*[D20] ZCL_CE_RAP_AGENCY_1234
ZCL_CE_RAP_AGENCY_1234
1  CLASS zcl_ce_rap_agency_1234 DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5
6    PUBLIC SECTION.
7      INTERFACES if_oo_adt_classrun.
8      INTERFACES if_rap
9
10     TYPES t_agency_ran
11     TYPES t_business_c
12
13     METHODS get_agenci
14       IMPORTING
15         filter_cond
16         top
17         skip                TYPE i OPTIONAL
18         is_data_requested   TYPE abap_bool
19         is_count_requested  TYPE abap_bool

1
2
if_rap_query_aggregation
if_rap_query_filter
if_rap_query_paging
if_rap_query_provider
if_rap_query_request
if_rap_query_response
Press 'Shift+Enter' to insert full signature
```

3. Ajoutez une implémentation pour la méthode **if_rap_query_provider~select**.

   - Vous pouvez utiliser la saisie semi-automatique du code **Ctrl+Espace** pour ajouter l'implémentation de la méthode **if_rap_query_provider~select**.

```
*[D20] ZCL_CE_RAP_AGENCY_1234
ZCL_CE_RAP_AGENCY_1234
1  CLASS zcl_ce_rap_agency_1234 DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5
6    PUBLIC SECTION.
7      INTERFACES if_oo_adt_classrun.
8      INTERFACES if_rap_query_provider.
9
10     TYPES t_ager
11     TYPES t_busi
12
13     METHODS get_
14       IMPORTING
15         filter_c
16         top
17         skip
18         is_data_
19         is_count
20       EXPORTING
21         business
22         count                    TYPE int8
23       RAISING
24         /iwbep/cx_cp_remote

Add implementation for select
Press 'Ctrl+Shift+1' to show in Quick Assist View
```

4. Mettre en œuvre la méthode **if_rap_query_provider~select**.

Dans la méthode select(), nous pouvons récupérer les détails de l'appel OData entrant à l'aide de l'objet **io_request**.

Cette méthode get_paging()permet de déterminer si une pagination côté client a été demandée lors de l'appel OData entrant. Elle get_filter()permet également de récupérer le filtre utilisé par la requête OData entrante et, en appelant la méthode ->get_as_ranges( )fournie par l'objet filtre, de récupérer le filtre sous forme de plages.

En utilisant les méthodes is_data_requested( )et, is_total_numb_of_rec_requested( )nous sommes en mesure de savoir si la demande entrante demande que des données commerciales soient renvoyées et si la demande contient un $count.

Cela s'avère utile pour que nous puissions fournir ces données lors de l'appel de la méthode get_agencies().

Remarque :
si des données commerciales sont demandées, il est obligatoire d'ajouter les données récupérées via la méthode set_data().
Si le nombre total d'entrées d'une entité est également demandé, le nombre d'entités renvoyées doit être défini via la méthode set_total_number_of_records().

Copiez et collez le code ci-dessous dans l'implémentation de la `METHOD if_rap_query_provider~select.`

```
    DATA business_data TYPE t_business_data.

    DATA(top)     = io_request->get_paging( )->get_page_size( ).

    DATA(skip)    = io_request->get_paging( )->get_offset( ).

    DATA(requested_fields)  = io_request->get_requested_elements( ).

    DATA(sort_order)    = io_request->get_sort_elements( ).

    DATA count TYPE int8.

    TRY.

        DATA(filter_condition) = io_request->get_filter( )->get_as_ranges( ).


        get_agencies(

                 EXPORTING

                   filter_cond        = filter_condition

                   top                = CONV i( top )

                   skip               = CONV i( skip )

                   is_data_requested  = io_request->is_data_requested( )

                   is_count_requested = io_request-
>is_total_numb_of_rec_requested(  )

                 IMPORTING

                   business_data  = business_data

                   count     = count

                 ) .
```

```
        IF io_request->is_total_numb_of_rec_requested(  ).
          io_response->set_total_number_of_records( count ).
        ENDIF.
        IF io_request->is_data_requested(  ).
          io_response->set_data( business_data ).
        ENDIF.


      CATCH cx_root INTO DATA(exception).
        DATA(exception_message) = cl_message_helper=>get_latest_t100_exception(
exception )->if_message~get_longtext( ).
    ENDTRY.
  ENDMETHOD.
```

5. Enregistrez et activez vos modifications.

Résumé

Vous avez implémenté la classe d'implémentation de requête de l'entité personnalisée. Nous pouvons ainsi améliorer l'implémentation de notre objet métier géré créé lors des semaines 2 et 3.

Nous utiliserons l'entité personnalisée

- Comme une aide de valeur
- Dans le cadre de la détermination de l'agence sélectionnée lors de la création ou de la modification des entités de voyage

Afin que vous ayez mis en œuvre un scénario d'extension côte à côte.

6.4 Amélioration du modèle de vol avec des données externes (Unité 4)

Objectif de l'unité

Améliorer le modèle de voyage de l'application gérée en y intégrant des données externes issues de l'entité personnalisée ZCE_RAP_AGENCY_####, via des ajustements dans :

- la vue interface du voyage,
- la vue de projection,
- l'implémentation du comportement.

Vérifier le code d'entité personnalisé

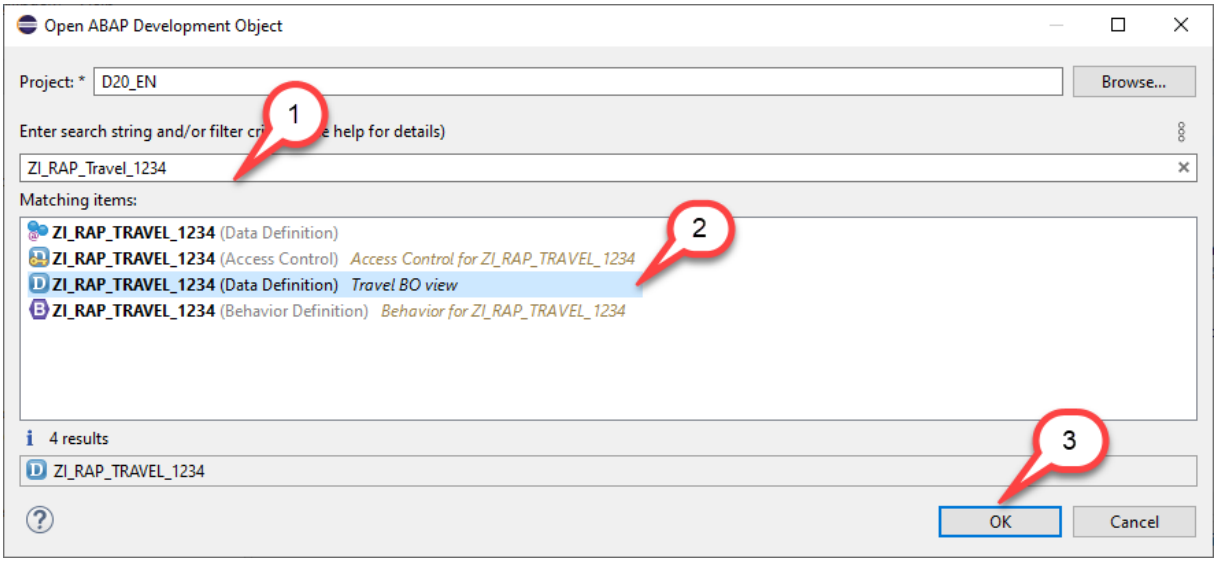
Commençons par examiner le code de notre entité personnalisée **zce_rap_agency_####** .

Veuillez noter que le champ clé s'appelle **AgencyId** avec un **d** minuscule. Comme nous ne pouvons pas créer d'alias pour les noms de champs, nous devons adapter la condition de l'aide à la valeur définie dans la vue d'interface, ainsi que la définition de l'aide à la valeur dans la vue de projection.

```
@EndUserText.label: 'agency data from ES5'

@ObjectModel.query.implementedBy: 'ABAP:ZCL_CE_RAP_AGENCY_####'

define custom entity zce_rap_agency_####

 {

 key AgencyId : abap.numc( 6 ) ;
```

Modifier le code de la vue de l'interface

1. Ouvrez la vue de l'interface CDS **ZI_RAP_Travel_####**de vos objets métier RAP gérés
    - Choisissez Ctrl+Maj+A
    - Entrez le nom de la vue de l'interface CDS **ZI_RAP_Travel_####**
    - Appuyez sur **OK**

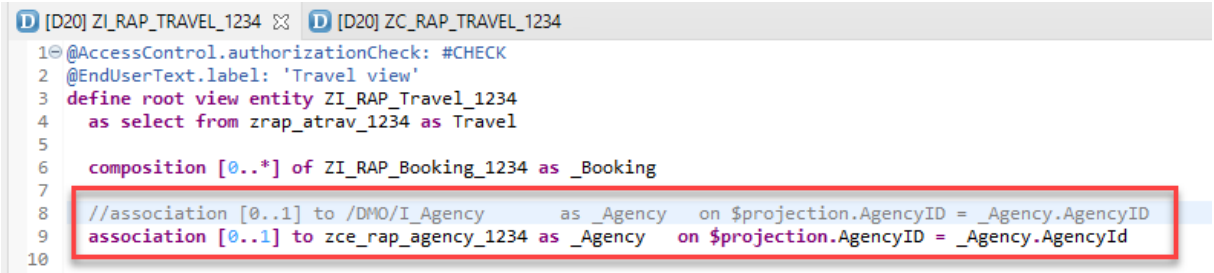


2. Changer le code de l'association **_Agency**.

Dans cette étape, nous modifions le code de l'association **_Agency** afin qu'elle n'utilise plus la vue CDS **/DMO/I_Agency** mais plutôt notre entité personnalisée nouvellement créée **zce_rap_agency_####**.

//association [0..1] à /DMO/I_Agency en tant que _Agency sur $projection.AgencyID = _Agency.AgencyID

association [0..1] à zce_rap_agency_#### en tant que _Agency sur $projection.AgencyID = _Agency.AgencyId



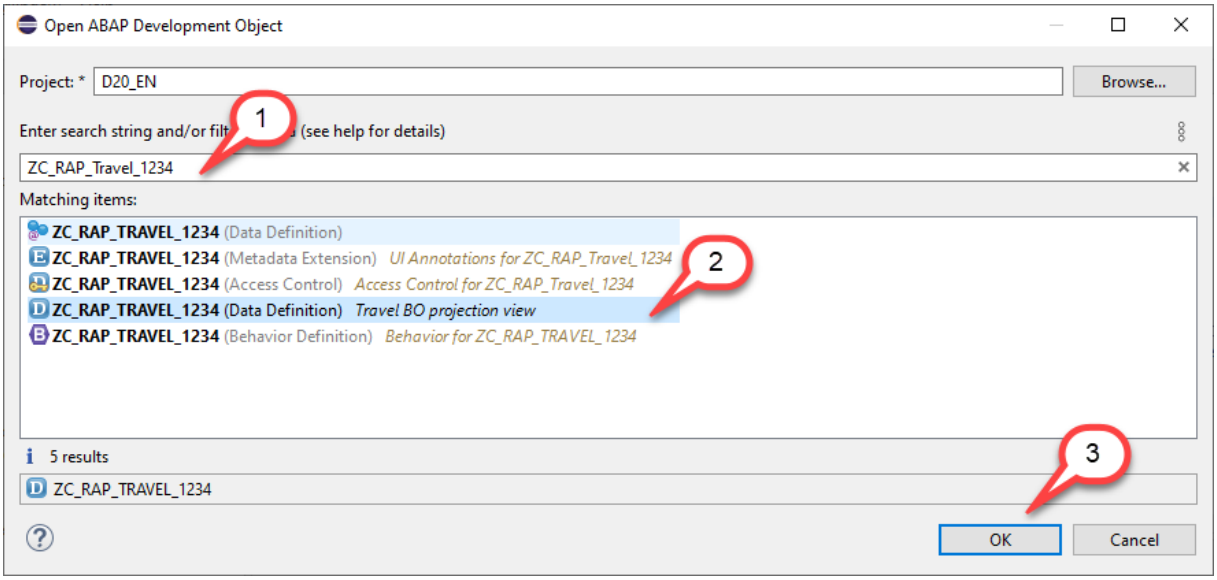
N'activez pas encore votre code.
Si vous essayez de le faire, vous obtiendrez le message d'erreur **« Association « *AGENCY » de l'entité « ZCE_RAP_AGENCY* #### » impossible à utiliser »** . En effet, l'association est toujours utilisée dans la vue de projection pour le champ **AgencyName** , lu via l'association _Agency.Name as AgencyName. Les entités personnalisées ne devant pas faire partie d'une projection, nous devons supprimer ce champ de la vue de projection, ce que nous ferons à l'étape suivante.

Problems | Properties | Templates | Bookmarks | Transport Organizer | Feed Reader | abapGit Repo

Message	Severity
==========	information
Activation of worklist (CB0000000022 / 10/18/2020 / 16:03:41 /)	information
==========	information
Technical log for mass activation	information
See log CB000000002220201018160340:ACT	information
DDLS ZI_RAP_TRAVEL_1234 activated	information
==========	information
Adjustment of active dependent objects	information
==========	information
DDLS ZC_RAP_TRAVEL_1234 is inconsistent in active version	error
Checks DDL Source ZC_RAP_TRAVEL_1234	information
Association "_AGENCY" of entity "ZCE_RAP_AGENCY_1234" cannot be used	error
DDL source ZC_RAP_TRAVEL_1234 contains errors	information
==========	information

Modifier le code de la vue de projection

1. Ouvrez la vue de l'interface CDS **ZC_RAP_Travel_####**de vos objets métier RAP gérés
   - Choisissez **Ctrl+Maj+A**
   - Entrez le nom de la vue de l'interface CDS**ZC_RAP_Travel_####**
   - Appuyez sur **OK**



2. Changer le code

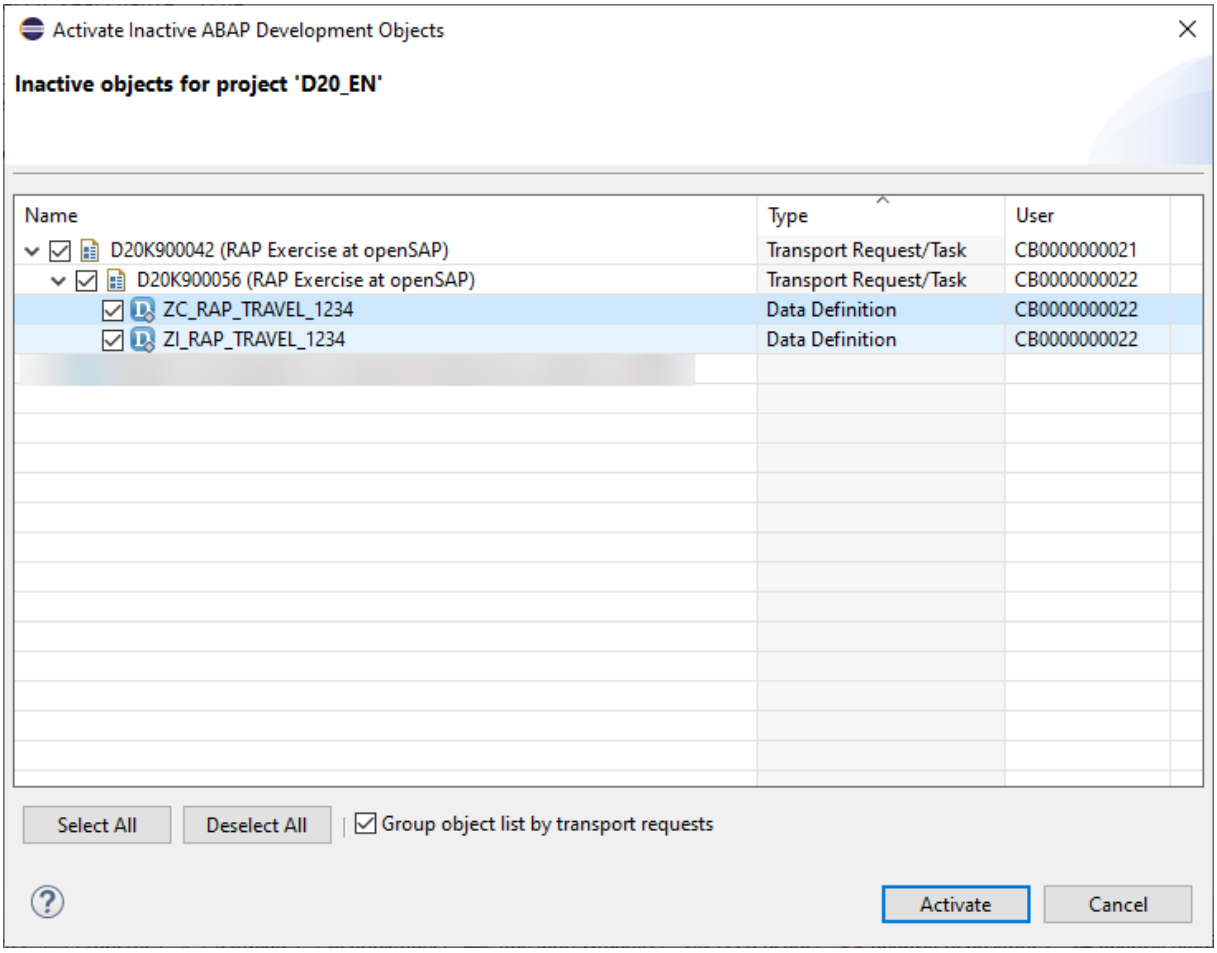
- Commentez le champ **AgencyName**car il utilise l'association **_Agency**qui n'est pas autorisée puisque les entités personnalisées ne doivent pas être utilisées dans les vues de projection
- Commentez l'annotation **@ObjectModel.text.element: ['AgencyName']**qui utilise le champ**AgencyName**
- Remplacez l'annotation de l'aide sur la valeur **AgencyID**afin que l'aide sur la valeur utilise l'entité personnalisée nouvellement créée au lieu de la vue CDS**/DMO/I_Agency**

```
[D20] ZI_RAP_TRAVEL_1234   *[D20] ZC_RAP_TRAVEL_1234
1  @EndUserText.label: 'Booking BO projection view'
2  @AccessControl.authorizationCheck: #CHECK
3  @Search.searchable: true
4  @Metadata.allowExtensions: true
5
6  define root view entity ZC_RAP_Travel_1234
7    as projection on ZI_RAP_Travel_1234 as Travel
8  {
9    key TravelUUID,
10       @Search.defaultSearchElement: true
11       TravelID,
12       //@Consumption.valueHelpDefinition: [{ entity: { name: '/DMO/I_Agency', element: 'AgencyID'} }]
13       @Consumption.valueHelpDefinition: [{ entity : {name: 'zce_rap_agency_1234', element: 'AgencyID' } }]
14       //@ObjectModel.text.element: ['AgencyName']
15       @Search.defaultSearchElement: true
16       AgencyID,
17       //_Agency.Name        as AgencyName,
18       @Consumption.valueHelpDefinition: [{ entity: { name: '/DMO/I_Customer', element: 'CustomerID'} }]
19       @ObjectModel.text.element: ['CustomerName']
20       @Search.defaultSearchElement: true
```

```
define root view entity ZC_RAP_Travel_1234

  as projection on ZI_RAP_Travel_1234 as Travel

{

  key TravelUUID,

      @Search.defaultSearchElement: true

      TravelID,

      //@Consumption.valueHelpDefinition: [{ entity: { name: '/DMO/I_Agency',
element: 'AgencyID'} }]

      @Consumption.valueHelpDefinition: [{ entity : {name: 'zce_rap_agency_####',
element: 'AgencyId' } }]

      //@ObjectModel.text.element: ['AgencyName']

      @Search.defaultSearchElement: true

      AgencyID,
```

```
    //_Agency.Name        as AgencyName,
```

3. Utilisez l'activation de masse pour activer vos modifications dans les deux vues CDS.
   - Appuyez sur **Ctrl+Maj+F3**
   - Sélectionnez les deux vues CDS



Modifier la classe d'implémentation du comportement

Ouvrez votre classe d'implémentation **ZBP_I_RAP_TRAVEL_####**que vous avez créée au cours de la semaine 3.

Au lieu de lire à partir de la table locale, **/dmo/agency**nous utilisons un appel distant au service OData dans ES5 en utilisant la méthode publique **zcl_ce_rap_agency_####( )->get_agencies()**dans notre requête personnalisée.

ZBP_I_RAP_TRAVEL_1234 ▸ LHC_TRAVEL ▸ VALIDATEAGENCY ▸ if

```
164  METHOD validateAgency.
165    " Read relevant travel instance data
166    READ ENTITIES OF zi_rap_travel_1234 IN LOCAL MODE
167      ENTITY Travel
168        FIELDS ( AgencyID ) WITH CORRESPONDING #( keys )
169      RESULT DATA(travels).
170
171    DATA agencies TYPE SORTED TABLE OF /dmo/agency WITH UNIQUE KEY agency_id.
172
173    " Optimization of DB select: extract distinct non-initial agency IDs
174    agencies = CORRESPONDING #( travels DISCARDING DUPLICATES MAPPING agency_id = AgencyID EXCEPT * ).
175    DELETE agencies WHERE agency_id IS INITIAL.
176
177 **    IF agencies IS NOT INITIAL.
178 **      " Check if agency ID exist
179 **      SELECT FROM /dmo/agency FIELDS agency_id
180 **        FOR ALL ENTRIES IN @agencies
181 **        WHERE agency_id = @agencies-agency_id
182 **        INTO TABLE @DATA(agencies_db).
183 **    ENDIF.
184
185
186    LOOP AT travels INTO DATA(travel).
187 *      " Clear state messages that might exist
188      APPEND VALUE #(  %tky               = travel-%tky
189                       %state_area        = 'VALIDATE_AGENCY' )
190        TO reported-travel.
191    ENDLOOP.
192
193    DATA filter_conditions  TYPE if_rap_query_filter=>tt_name_range_pairs .
194    DATA ranges_table TYPE if_rap_query_filter=>tt_range_option .
195    DATA business_data TYPE TABLE OF zrap_1234z_travel_agency_es5.
196
197    IF  agencies IS NOT INITIAL.
198
199      ranges_table = VALUE #( FOR agency IN agencies (  sign = 'I' option = 'EQ' low = agency-agency_id ) ).
200      filter_conditions = VALUE #( ( name = 'AGENCYID'  range = ranges_table ) ).
201
202
203
204      TRY.
205          "skip and top must not be used
206          "but an appropriate filter will be provided
207          NEW zcl_ce_rap_agency_1234( )->get_agencies(
208            EXPORTING
209              filter_cond     = filter_conditions
210              is_data_requested  = abap_true
211              is_count_requested = abap_false
212            IMPORTING
213              business_data  = business_data
214            ) .
215
216        CATCH /iwbep/cx_cp_remote
217              /iwbep/cx_gateway
218              cx_web_http_client_error
219              cx_http_dest_provider_error
220
221       INTO DATA(exception).
222
```

Plus tard dans l'implémentation, nous remplaçons également la vérification des entrées dans la table **agency_db**par une vérification des entrées dans la table interne **business_data**qui est renvoyée en appelant la méthode **zcl_ce_rap_agency_####( )->get_agencies( )**.

```
242
243     " Raise msg for non existing and initial agencyID
244 **    LOOP AT travels INTO DATA(travel).
245     LOOP AT travels INTO travel.
246
247
248 **      " Clear state messages that might exist
249 **      APPEND VALUE #(  %tky               = travel-%tky
250 **                       %state_area        = 'VALIDATE_AGENCY' )
251 **        TO reported-travel.
252
253
254
255 **      IF travel-AgencyID IS INITIAL OR NOT line_exists( agencies_db[ agency_id = travel-AgencyID ] ).
256       IF travel-AgencyID IS INITIAL OR NOT line_exists( business_data[ agencyid = travel-AgencyID ] ).
257
258         APPEND VALUE #( %tky = travel-%tky ) TO failed-travel.
259
260         APPEND VALUE #( %tky        = travel-%tky
261                         %state_area = 'VALIDATE_AGENCY'
262                         %msg        = NEW zcm_rap_1234(
263                                           severity = if_abap_behv_message=>severity-error
264                                           textid   = zcm_rap_1234=>agency_unknown
265                                           agencyid = travel-AgencyID )
266                         %element-AgencyID = if_abap_behv=>mk-on )
267           TO reported-travel.
268       ENDIF.
269     ENDLOOP.
270   ENDMETHOD.
271
```

Votre code devrait donc ressembler à ce qui suit

```
METHOD validateAgency.

    " Read relevant travel instance data

    READ ENTITIES OF zi_rap_travel_#### IN LOCAL MODE

      ENTITY Travel

        FIELDS ( AgencyID ) WITH CORRESPONDING #( keys )

      RESULT DATA(travels).


    DATA agencies TYPE SORTED TABLE OF /dmo/agency WITH UNIQUE KEY agency_id.


    " Optimization of DB select: extract distinct non-initial agency IDs

    agencies = CORRESPONDING #( travels DISCARDING DUPLICATES MAPPING agency_id =
AgencyID EXCEPT * ).

    DELETE agencies WHERE agency_id IS INITIAL.


**    IF agencies IS NOT INITIAL.
```

```
**      " Check if agency ID exist

**      SELECT FROM /dmo/agency FIELDS agency_id

**        FOR ALL ENTRIES IN @agencies

**        WHERE agency_id = @agencies-agency_id

**        INTO TABLE @DATA(agencies_db).

**    ENDIF.




    LOOP AT travels INTO DATA(travel).

*      " Clear state messages that might exist

      APPEND VALUE #(  %tky               = travel-%tky

                       %state_area        = 'VALIDATE_AGENCY' )

        TO reported-travel.

    ENDLOOP.


    DATA filter_conditions  TYPE if_rap_query_filter=>tt_name_range_pairs .

    DATA ranges_table TYPE if_rap_query_filter=>tt_range_option .

    DATA business_data TYPE TABLE OF zrap_####z_travel_agency_es5.


    IF  agencies IS NOT INITIAL.


      ranges_table = VALUE #( FOR agency IN agencies (  sign = 'I' option = 'EQ'
low = agency-agency_id ) ).

      filter_conditions = VALUE #( ( name = 'AGENCYID'  range = ranges_table ) ).
```

```
      TRY.

          "skip and top must not be used

          "but an appropriate filter will be provided

          NEW zcl_ce_rap_agency_1234( )->get_agencies(

            EXPORTING

              filter_cond    = filter_conditions

              is_data_requested  = abap_true

              is_count_requested = abap_false

            IMPORTING

              business_data  = business_data

            ) .


        CATCH /iwbep/cx_cp_remote

              /iwbep/cx_gateway

              cx_web_http_client_error

              cx_http_dest_provider_error


       INTO DATA(exception).


          DATA(exception_message) = cl_message_helper=>get_latest_t100_exception(
exception )->if_message~get_text( ) .


          LOOP AT travels INTO travel.

            APPEND VALUE #( %tky = travel-%tky ) TO failed-travel.


            APPEND VALUE #( %tky        = travel-%tky

                            %state_area = 'VALIDATE_AGENCY'
```

```
                            %msg        =  new_message_with_text( severity =
if_abap_behv_message=>severity-error text = exception_message )

                            %element-AgencyID = if_abap_behv=>mk-on )

              TO reported-travel.

          ENDLOOP.



          RETURN.



      ENDTRY.



    ENDIF.





    " Raise msg for non existing and initial agencyID

**    LOOP AT travels INTO DATA(travel).

    LOOP AT travels INTO travel.




**      " Clear state messages that might exist

**      APPEND VALUE #(  %tky               = travel-%tky

**                       %state_area        = 'VALIDATE_AGENCY' )

**        TO reported-travel.
```

```
**      IF travel-AgencyID IS INITIAL OR NOT line_exists( agencies_db[ agency_id =
travel-AgencyID ] ).

      IF travel-AgencyID IS INITIAL OR NOT line_exists( business_data[ agencyid =
travel-AgencyID ] ).



        APPEND VALUE #( %tky = travel-%tky ) TO failed-travel.



        APPEND VALUE #( %tky        = travel-%tky

                        %state_area = 'VALIDATE_AGENCY'

                        %msg        = NEW zcm_rap_####(

                                            severity =
if_abap_behv_message=>severity-error

                                            textid   = zcm_rap_####=>agency_unknown

                                            agencyid = travel-AgencyID )

                        %element-AgencyID = if_abap_behv=>mk-on )

          TO reported-travel.

      ENDIF.

    ENDLOOP.

  ENDMETHOD.
```

Résumé

Vous avez amélioré la mise en œuvre de votre objet métier géré que nous avons créé au cours des semaines 2 et 3, de telle sorte que

- L’entité personnalisée peut être utilisée comme aide à la valeur et
- La validation pour les agences utilise également la même implémentation que l'entité personnalisée pour récupérer les agences de voyages du système distant.

En conséquence, vous avez mis en œuvre un scénario d’extension côte à côte.

6.5 Création et aperçu du service d'interface utilisateur OData (Unité 5)

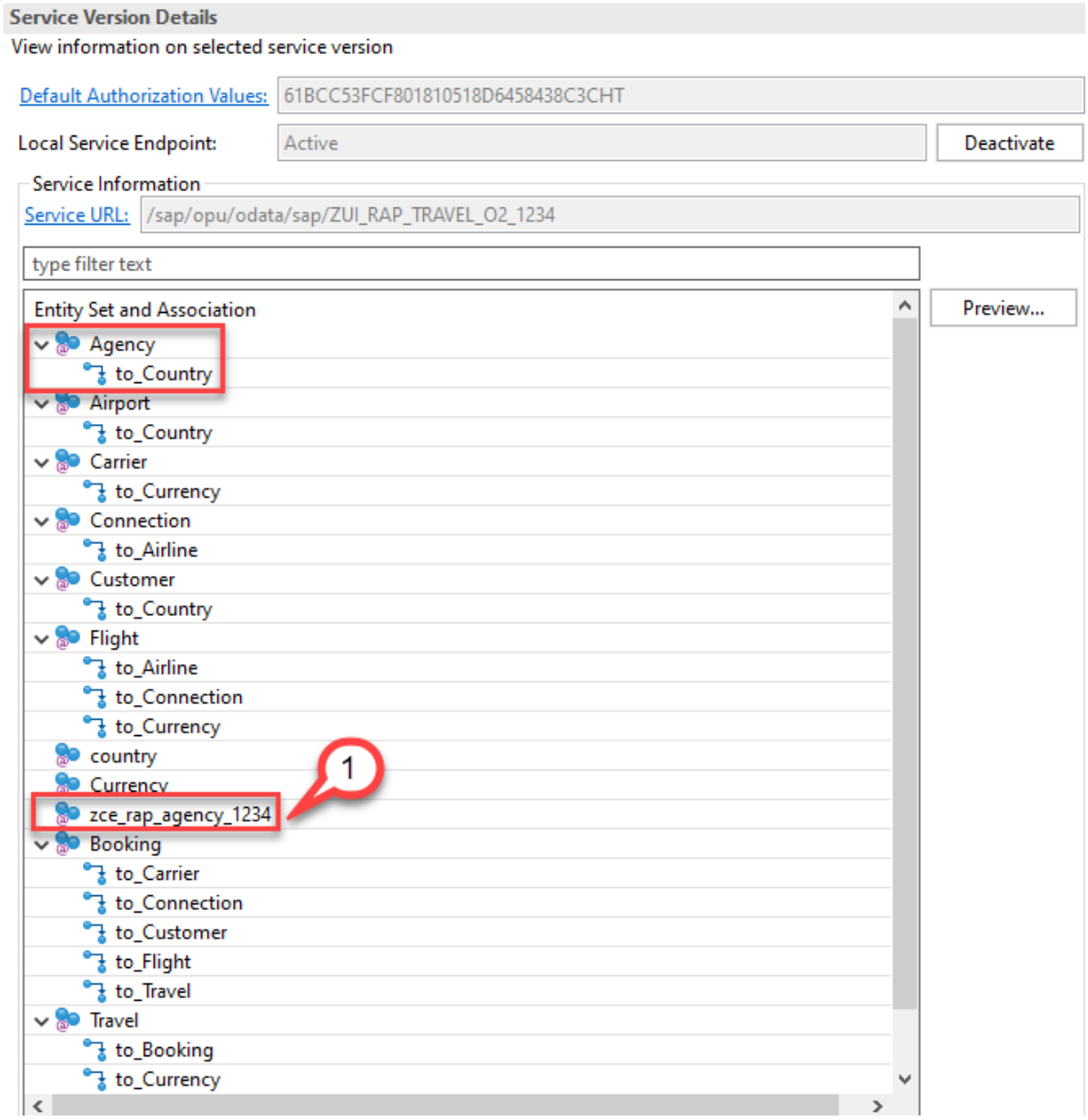
Objectif de l'unité

Prévisualiser le service OData UI final de l'application.
Ce service intègre désormais des données d'agences issues du système distant ES5, visibles via l'aide à la saisie lors de la création de voyages.

Vérifiez la liaison du service

1. Développez votre forfait **ZRAP_TRAVEL_#### > Services aux entreprises > Liaisons de services**
   - Ouvrir la liaison de serviceZUI_RAP_TRAVEL_O2_####

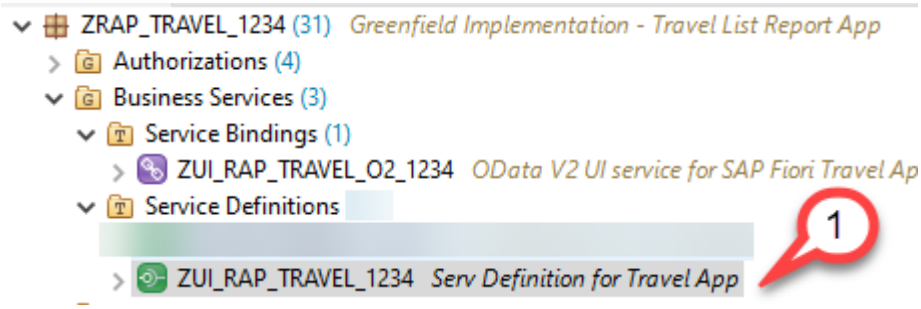
2. Recherchez l'entité personnalisée **zce_rap_agency_####** dans la liste des ensembles d'entités et des associations.



En examinant de plus près la liaison de service, nous constatons que l'entité personnalisée est répertoriée dans la liste des ensembles et associations d'entités. L'entité personnalisée a été automatiquement ajoutée à la liste des entités, car elle est utilisée comme aide à la saisie dans la vue de projection. Cependant, elle est répertoriée avec son nom technique ; nous allons donc modifier cela en l'ajoutant à la définition de service.

Modifier la définition du service

1. Ouvrir la définition du service

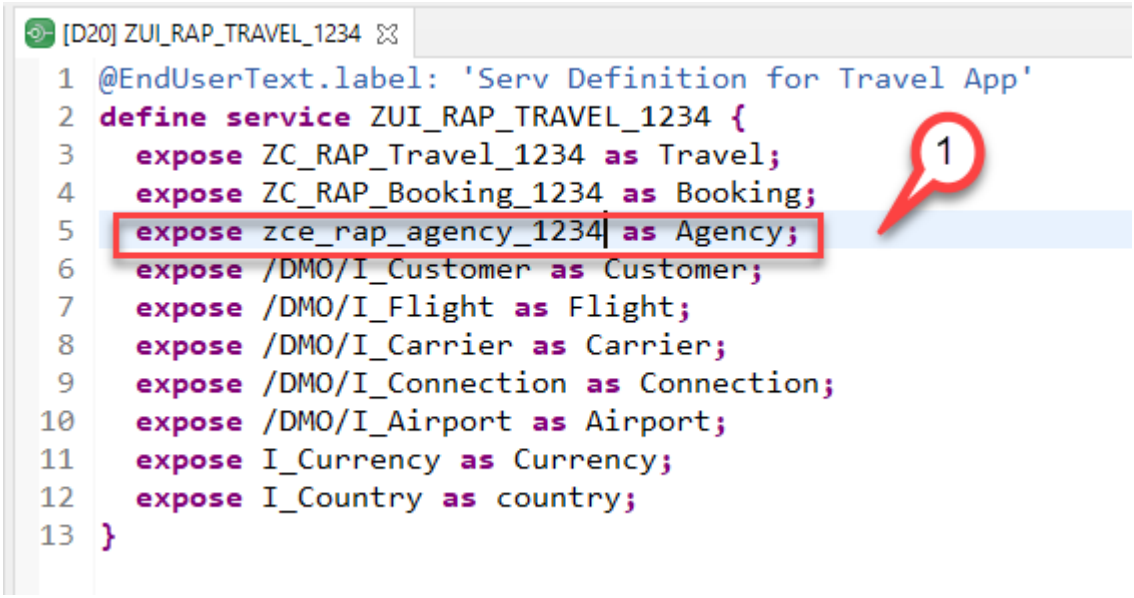


2. Modifiez l'instruction d'exposition afin que la vue cds **/DMO/I_Agency** ne soit plus exposée en tant que Travel mais l'entité personnalisée **zce_rap_agency_####** .
   - **Remplacer** l'instruction d'exposition existante

```
expose /DMO/I_Agency as Agency;
```

avec la déclaration d'exposition suivante

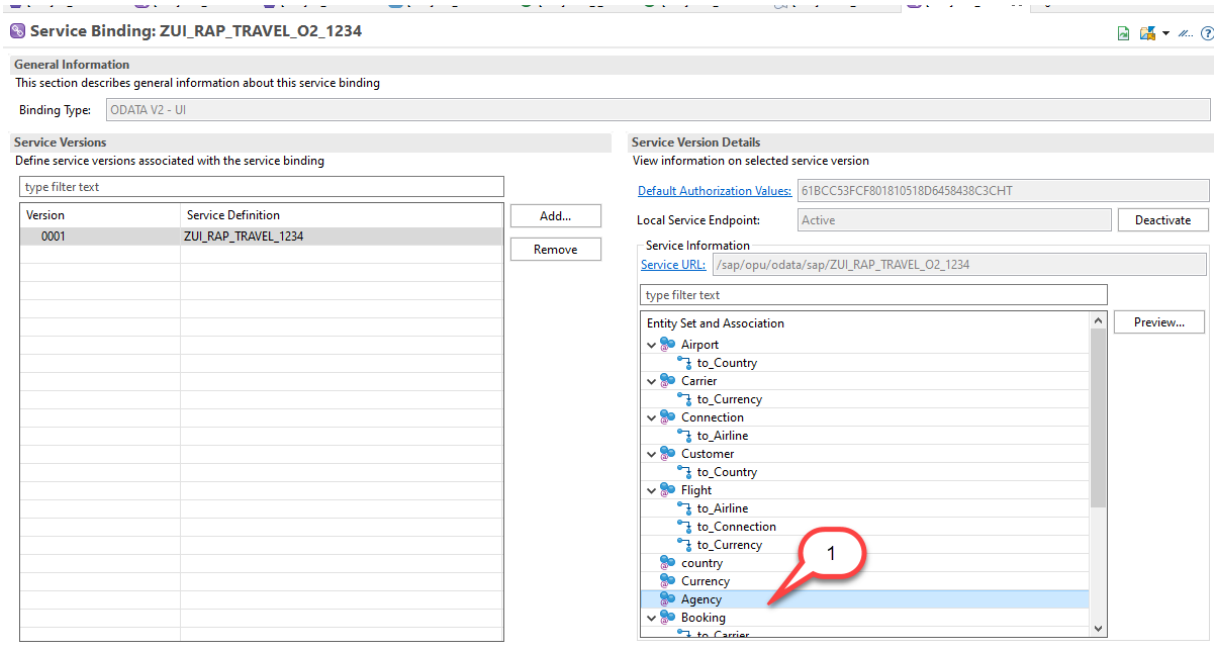
- `expose zce_rap_agency_#### as Agency;`

3. 

4. Veuillez noter qu'il est impossible d'ajouter des commentaires au code source de votre définition de service. Il est donc impossible de commenter l'instruction existante.
5. Exposer /DMO/I_Agency comme agence
6. Il doit plutôt être supprimé et remplacé par l'extrait de code mentionné ci-dessus.
7. Activez vos modifications

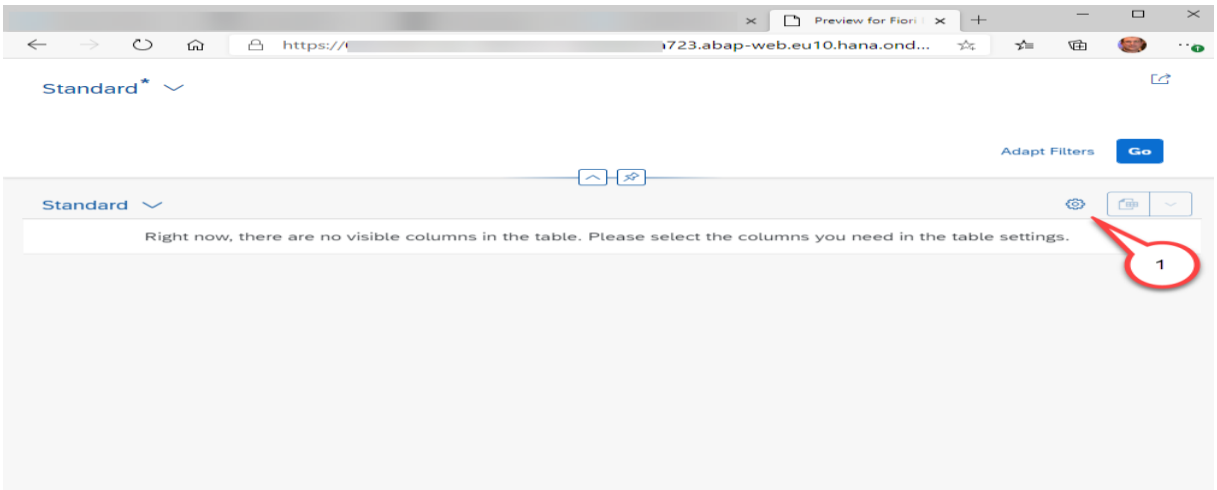
Tester l'accès à l'entité Agence

1. Revenir à la liaison de serviceZUI_RAP_TRAVEL_O2_####



2. Utilisez **Actualiser** dans le menu contextuel pour mettre à jour la liste des ensembles d'entités

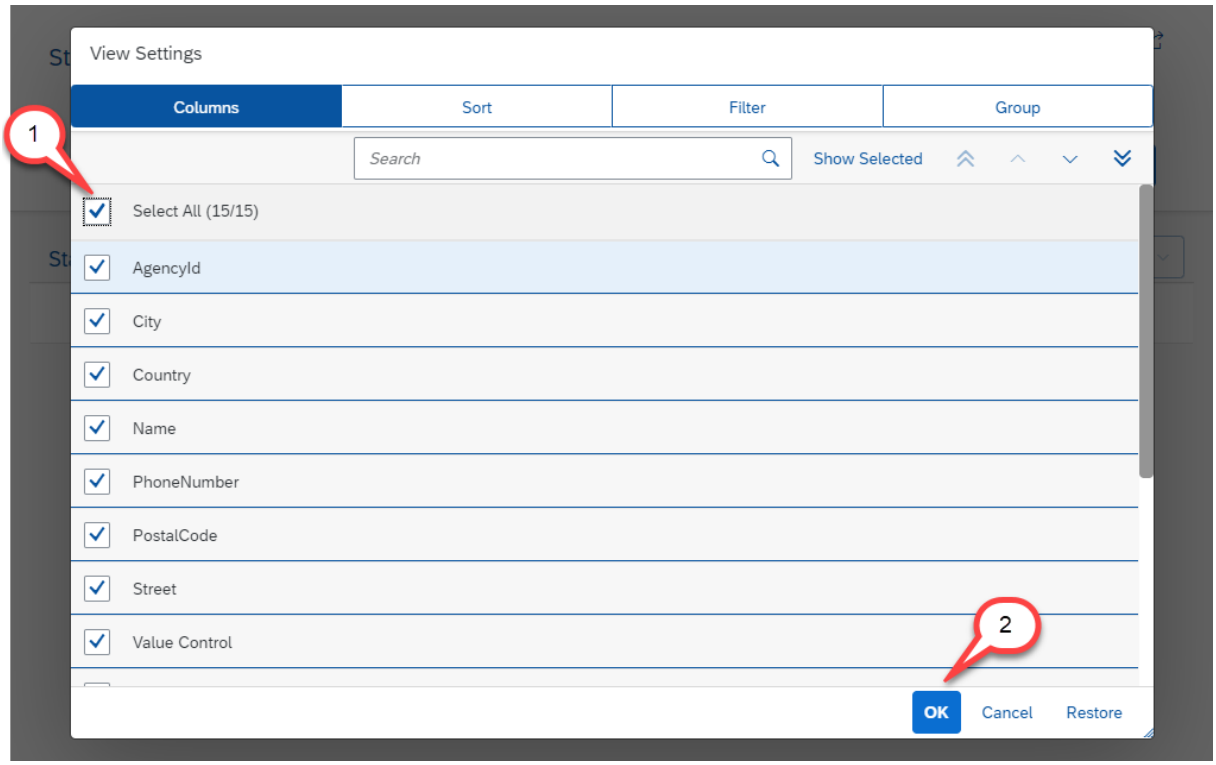
3. Double-cliquez sur l'entité Agency



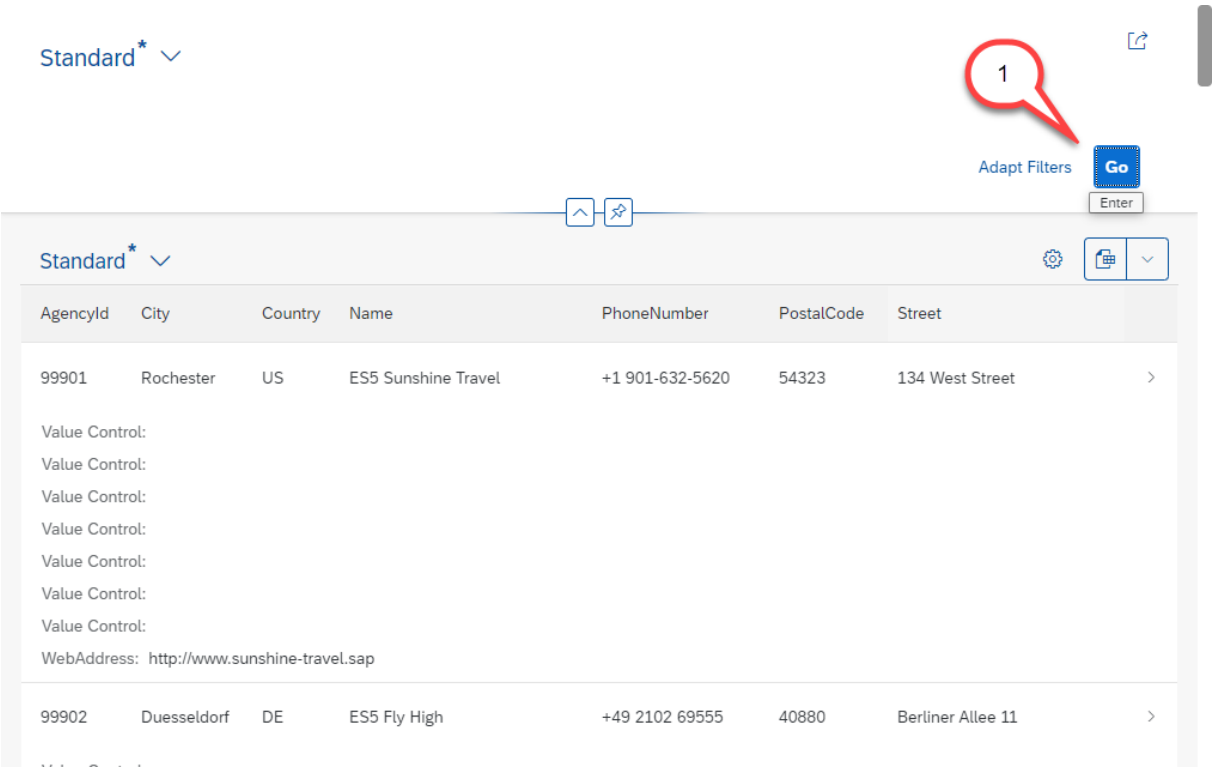
4. Vous remarquerez qu'aucune colonne n'est affichée, car aucune annotation d'interface utilisateur n'a été ajoutée à l'entité personnalisée. Utilisez l'icône *d'engrenage* pour sélectionner toutes les colonnes.

   - Cliquez sur l'icône **d'engrenage**

- Cliquez sur la case à cocher **Sélectionner tout**
- Appuyez sur **OK**



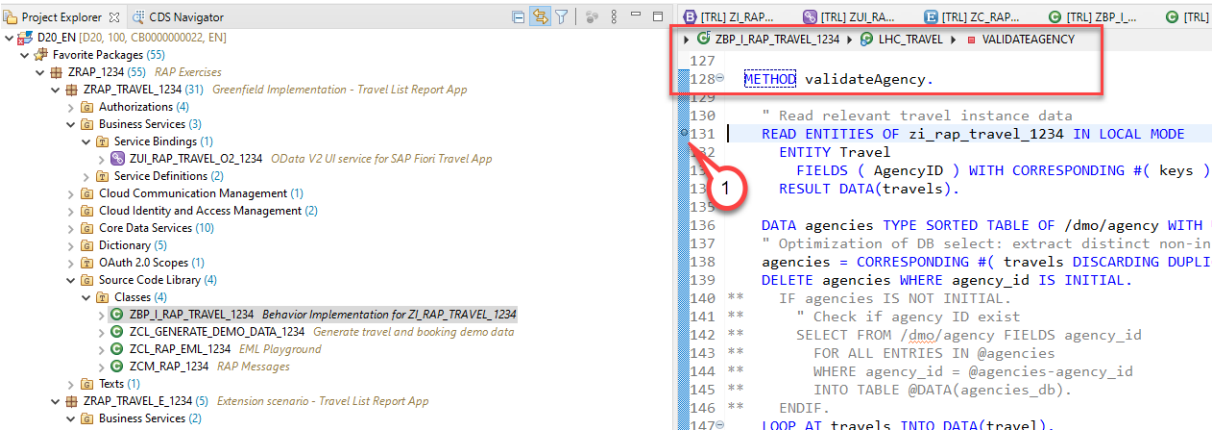
5. Appuyez sur **Go**



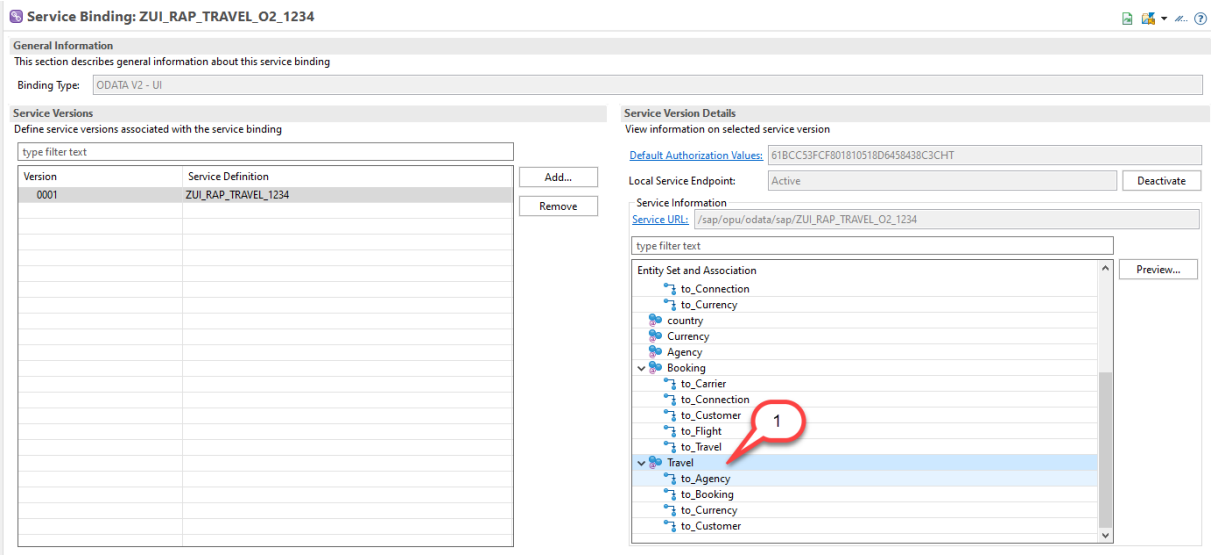
Cela récupérera une liste d'agences du système distant indiquées par les différents numéros d'agence et en ajoutant « ES5 » à leur nom

Ensemble d'entités de test Voyage

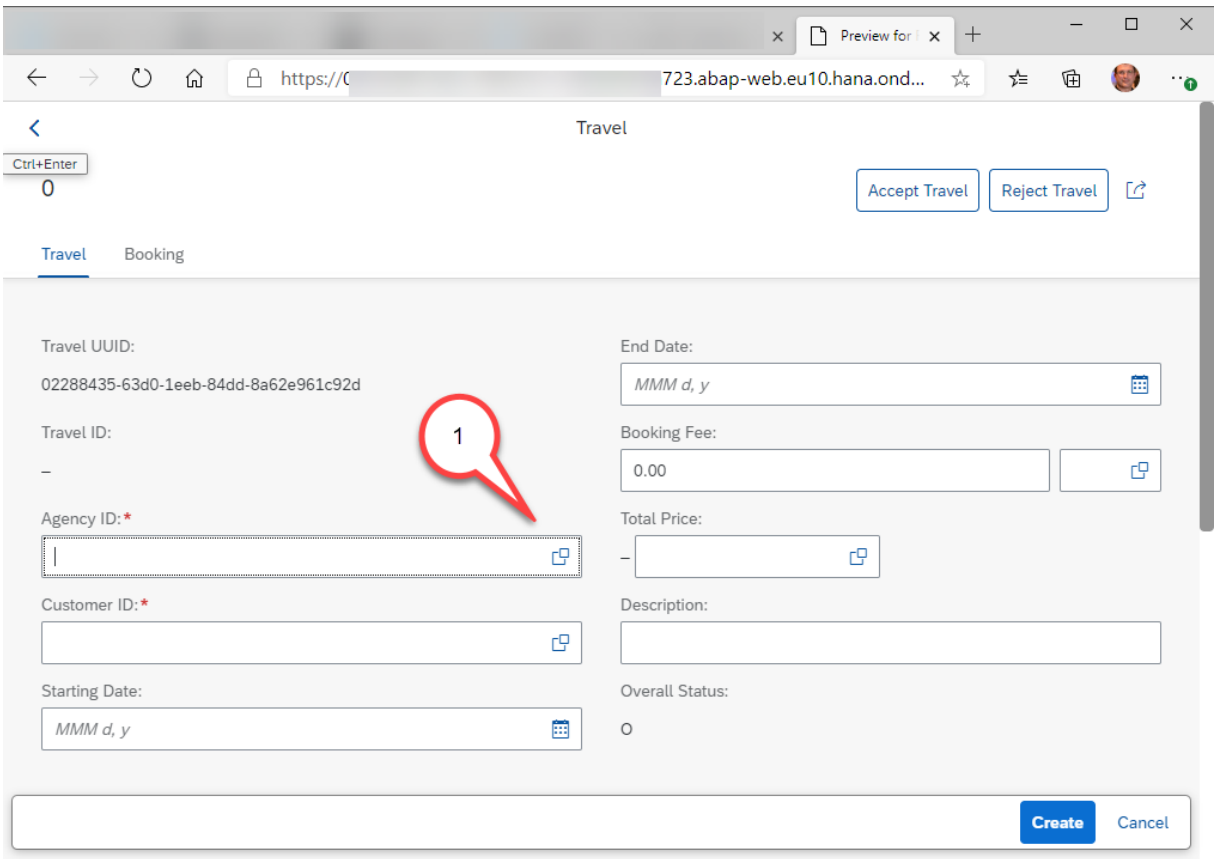
1. Créer un point d'arrêt dans la méthode validateAgency de la classe

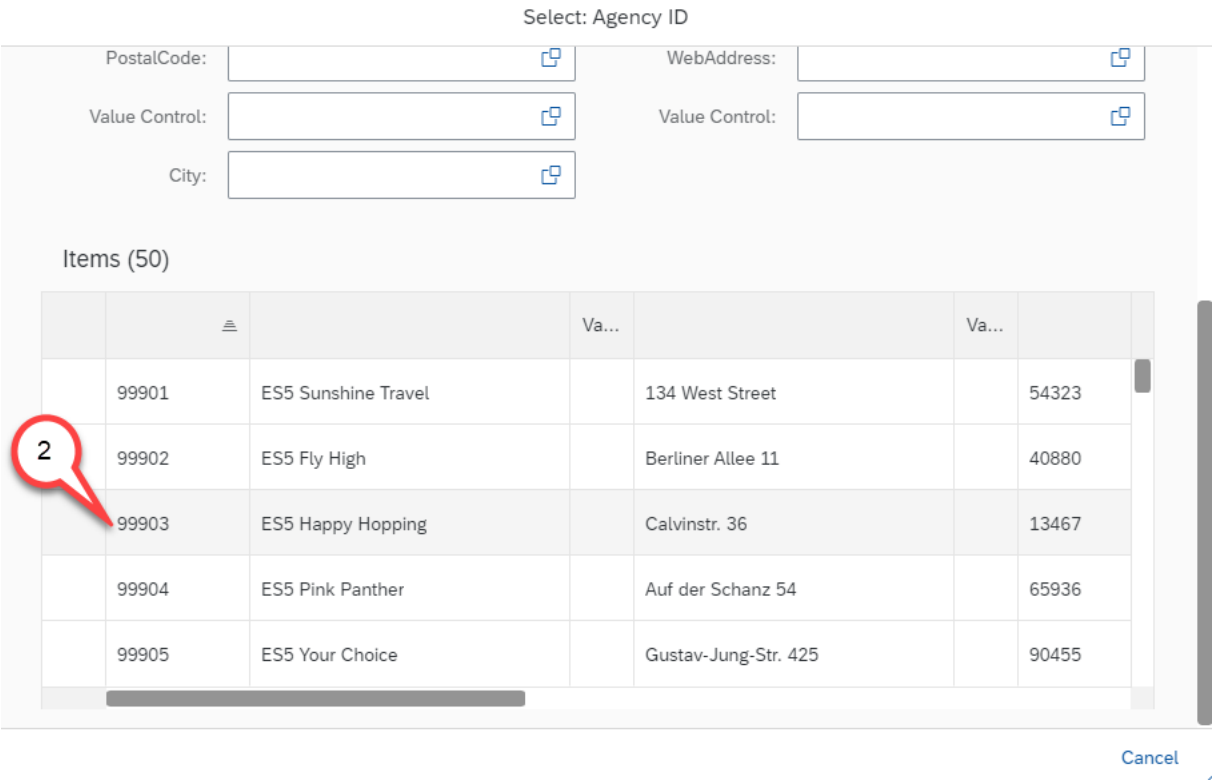


2. Revenez à la liaison de service et double-cliquez sur l'ensemble d'entités **Travel** pour démarrer l'aperçu de Fiori Elements



3. Créer une nouvelle entité de voyage
   - Lorsque vous utilisez l'aide à la valeur, vous verrez que les données de l'agence sont récupérées via l'entité personnalisée





- Lorsque vous ajoutez un point d'arrêt dans la validation, vous remarquerez que la validation est appelée trois fois si le brouillon est utilisé et deux fois sans brouillon.

Résumé

Dans cette unité, vous avez appris à adapter la définition de service et à savoir que ces modifications deviennent immédiatement actives dans la liaison de service publiée.

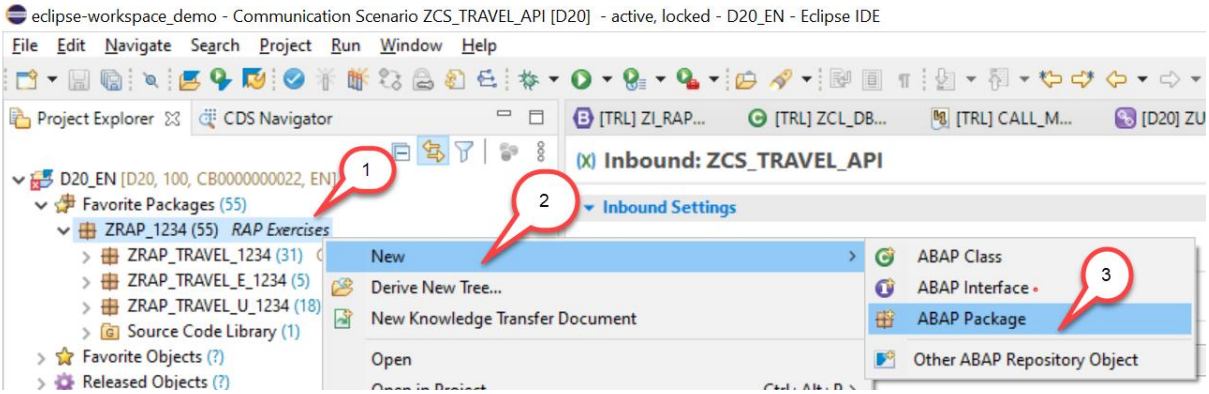
6.6 Création et aperçu de l'API Web OData (Unité 6)

Objectif de l'unité

Créer une **Web API** avec le **modèle RAP** et tester son accès externe via un outil comme **REST Client**.

Créer un nouveau package

1. Nous allons dans l'explorateur de projets, faisons un clic droit sur notre package **ZRAP_####**
2. ***Nouveau > Package ABAP*** .



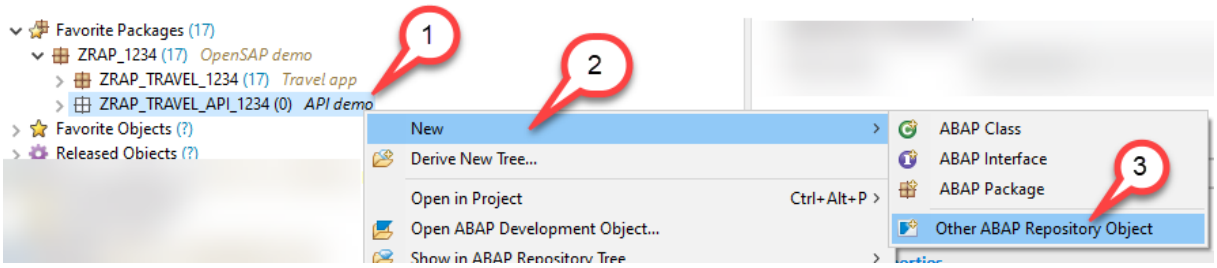
3. Conservez **ZRAP_TRAVEL_API_####**le nom (où ####se trouve le suffixe choisi et où _API indique qu'il s'agit d'un scénario d'API Web) et une description pertinente (par exemple, **scénario d'API de voyage**), puis cliquez sur **Suivant** pour continuer.
   Les champs Projet et Superpackage sont automatiquement attribués.

4. Sélectionnez une demande de transport existante ou créez-en une nouvelle, puis choisissez **Terminer** pour créer le nouveau package.

Créer une définition de service

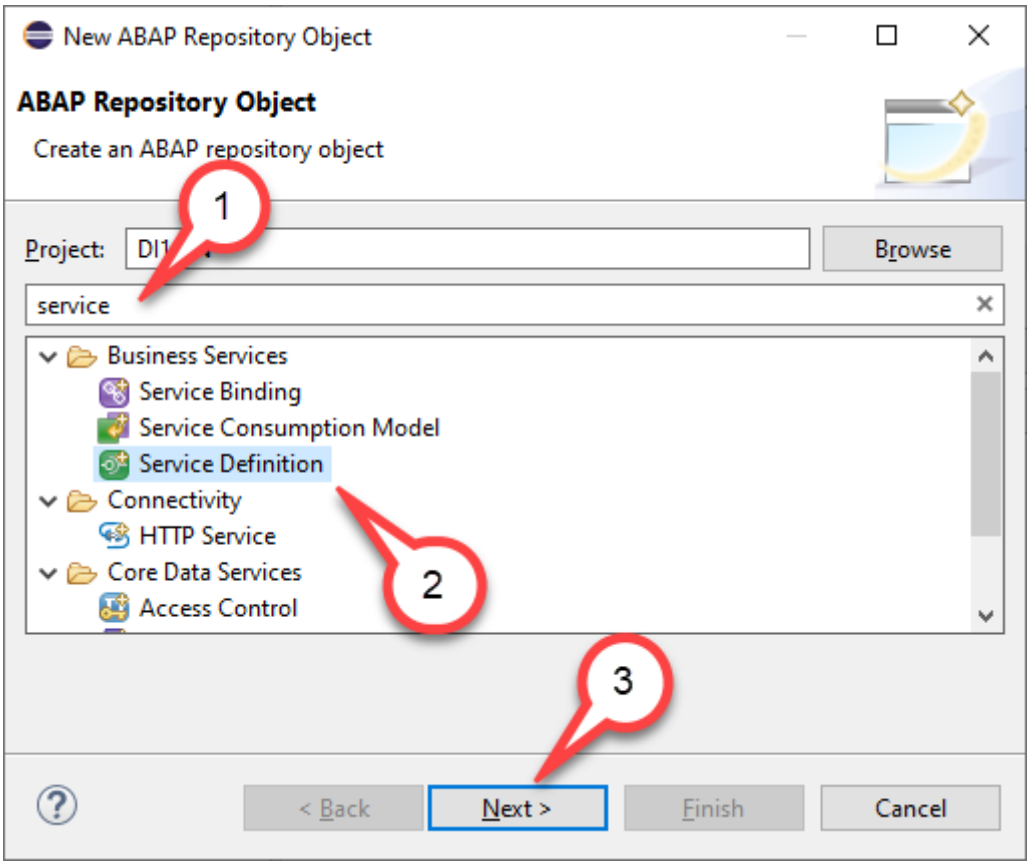
Commençons par créer une nouvelle définition de service, car le service de notre API Web n'a pas besoin de publier toutes les entités de notre service OData. Selon les besoins, vous pouvez également créer une couche de projection distincte pour une API Web, car c'est là que vous pourrez spécifier plus précisément les types d'opérations à prendre en charge par votre service. Pour simplifier le scénario de ce cours OpenSAP, nous allons simplement créer une nouvelle définition de service et une nouvelle liaison de service. C'est parti !

1. Faites un clic droit sur le package et sélectionnez **Nouveau > Autre objet de référentiel ABAP**



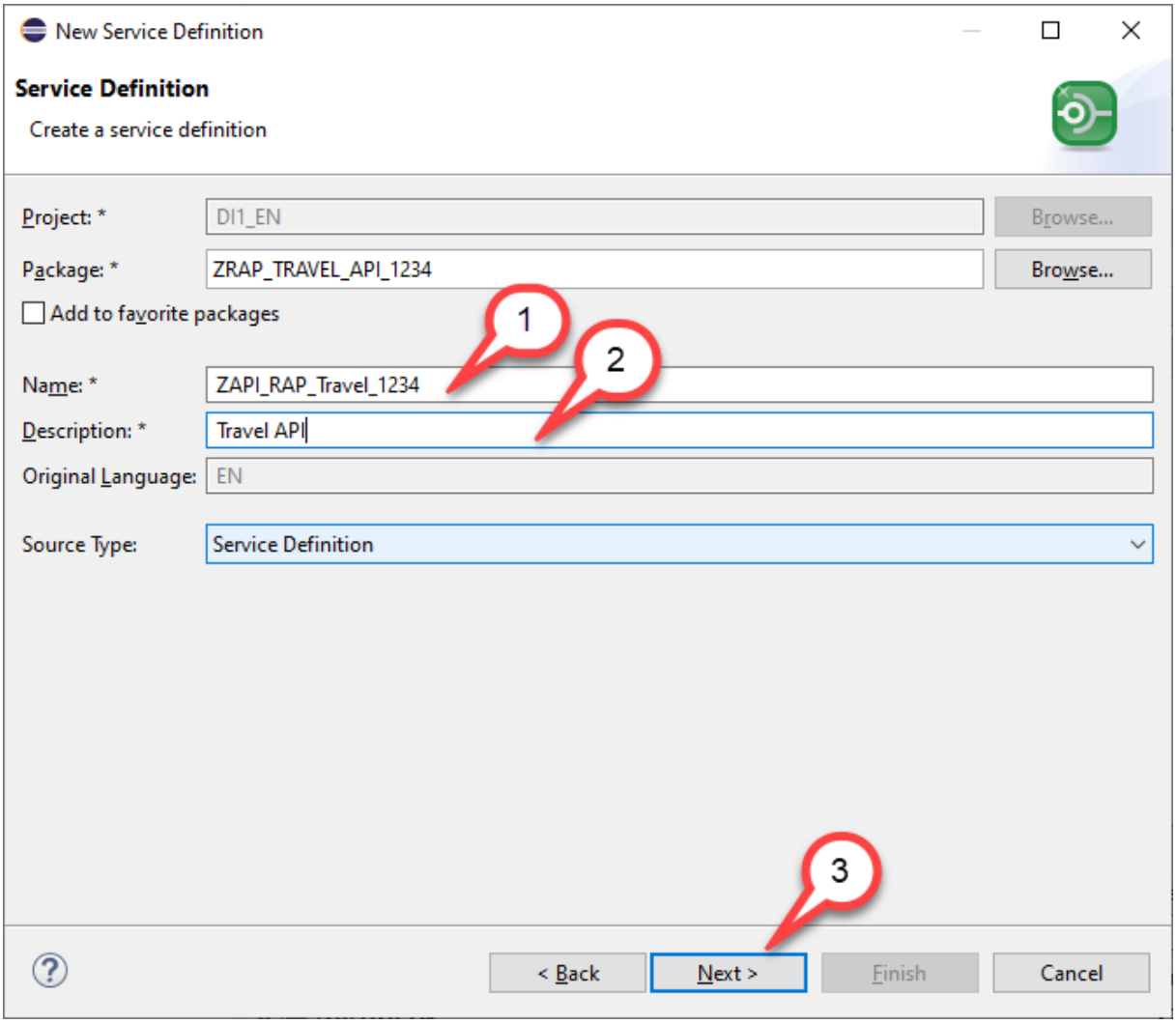
2. Dans la boîte de dialogue **Nouvel objet de référentiel ABAP**
   - Commencez à taper Servicedans la zone de texte
   - Sélectionnez Service Defintiondans la liste des objets trouvés

Appuyez **sur Suivant**

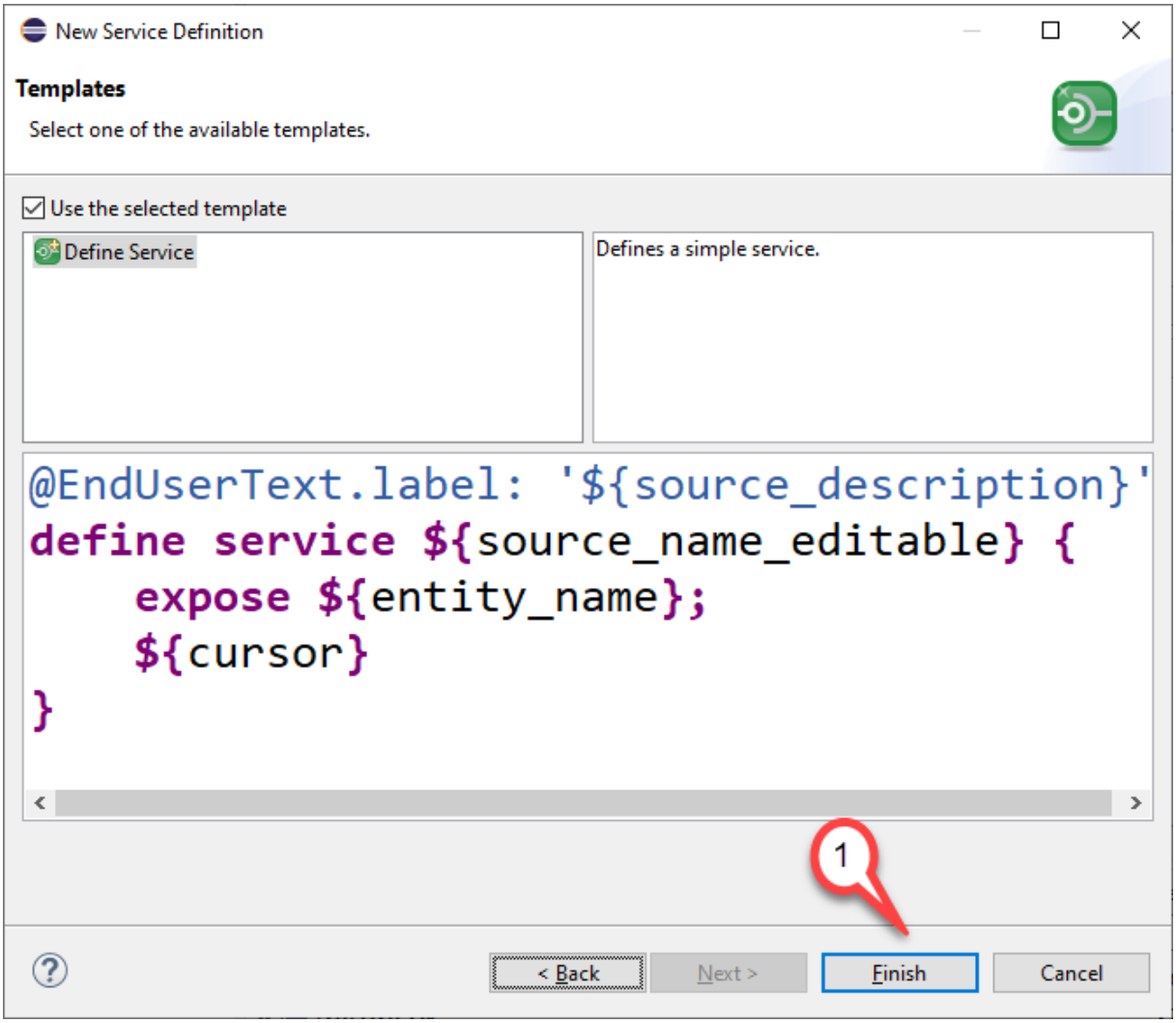


3. Entrez les valeurs suivantes
   - Nom:ZAPI_RAP_Travel_####
   - Description :Travel API

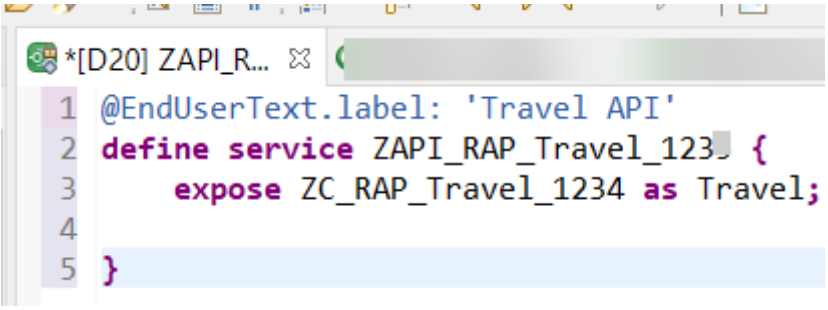
Appuyez **sur Suivant**



4. Sélection d'une demande de transport
   - Sélectionnez une demande de transport
   - Appuyez **sur Suivant**
5. Dans la boîte de dialogue **Nouvelle définition de service,** appuyez simplement sur **Terminer**



6. Dans l'éditeur de définition de service, nous n'ajouterons aucune entité supplémentaire car pour notre démonstration, il suffira de publier uniquement l'entité qui contient les données de voyage.



```
@EndUserText.label: 'API scenario'

define service ZAPI_RAP_Travel_#### {

    expose ZC_RAP_Travel_#### as Travel;



}
```

5. Activez vos modifications

Créer une liaison de service

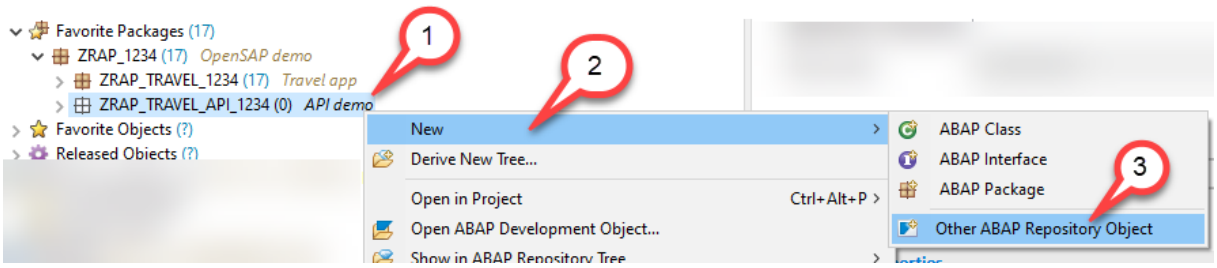
Nous allons maintenant créer une liaison de service pour notre API Web, nommée ZAPI_RAP_TRAVEL_####. Assurez-vous de sélectionner « ODATA V2 – Web API » comme type de liaison. La définition de service utilisée ici est ZAPI_RAP_TRAVEL_U_####.

Veuillez noter que lorsque nous utilisons la vue de consommation, les informations de valeur sont également publiées. En revanche, les métadonnées supplémentaires nécessaires à l'utilisation du service via Fiori Elements ne sont pas publiées.

Si vous souhaitez supprimer la publication des aides à la valeur, vous devez spécifier ce comportement dans la couche de projection en créant des vues de projection qui ne contiennent pas l'annotation @Consumption.valueHelpDefinition

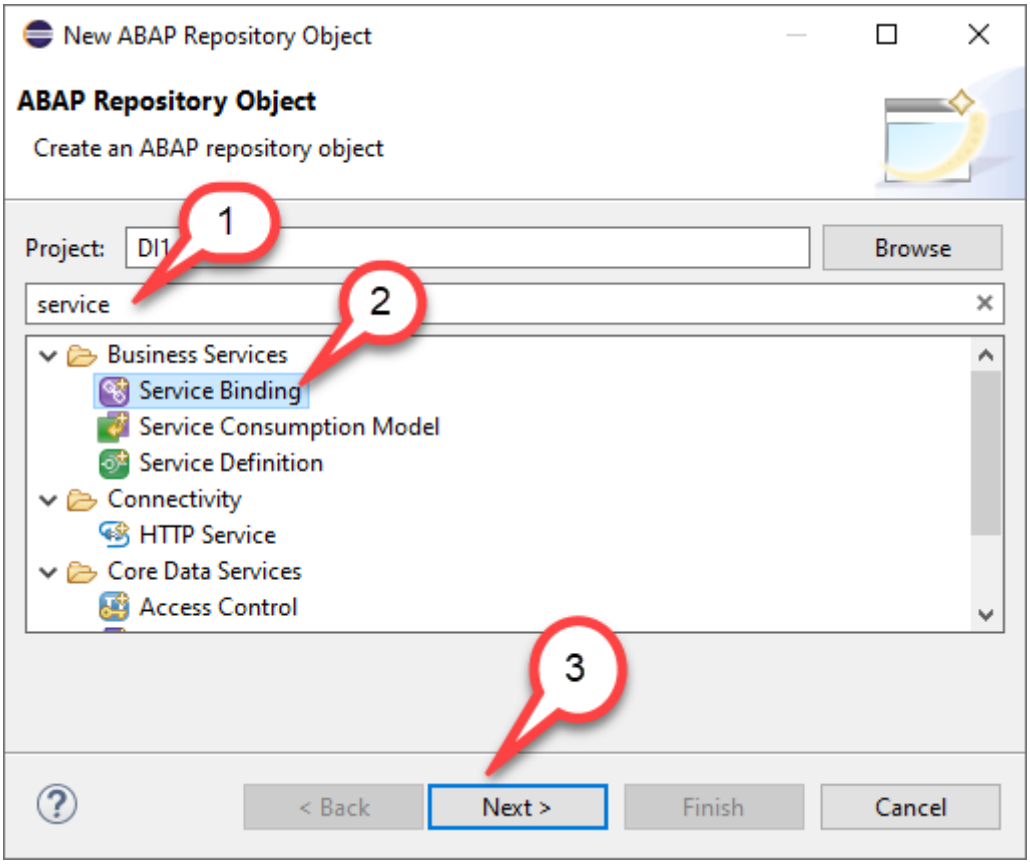
La publication automatique des aides à la valeur dans les API Web sera supprimée car il s'agit d'un bug qui était présent au moment de la rédaction de ce tutoriel

1. Faites un clic droit sur le package et sélectionnez **Nouveau > Autre objet de référentiel ABAP**



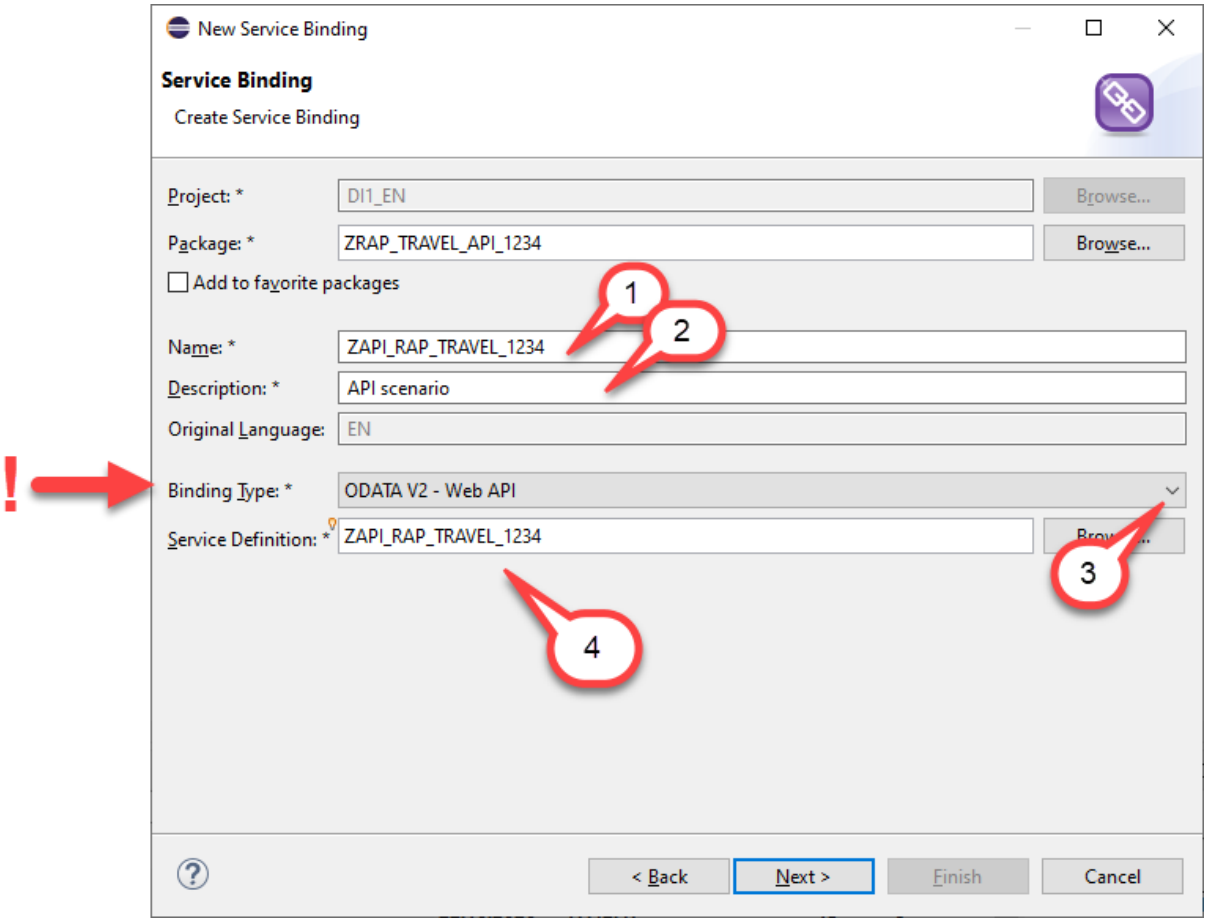
2. Dans la boîte de dialogue **Nouvel objet de référentiel ABAP**
   - Commencez à taper Servicedans la zone de texte
   - Sélectionnez Service Bindingdans la liste des objets trouvés

Appuyez **sur Suivant**

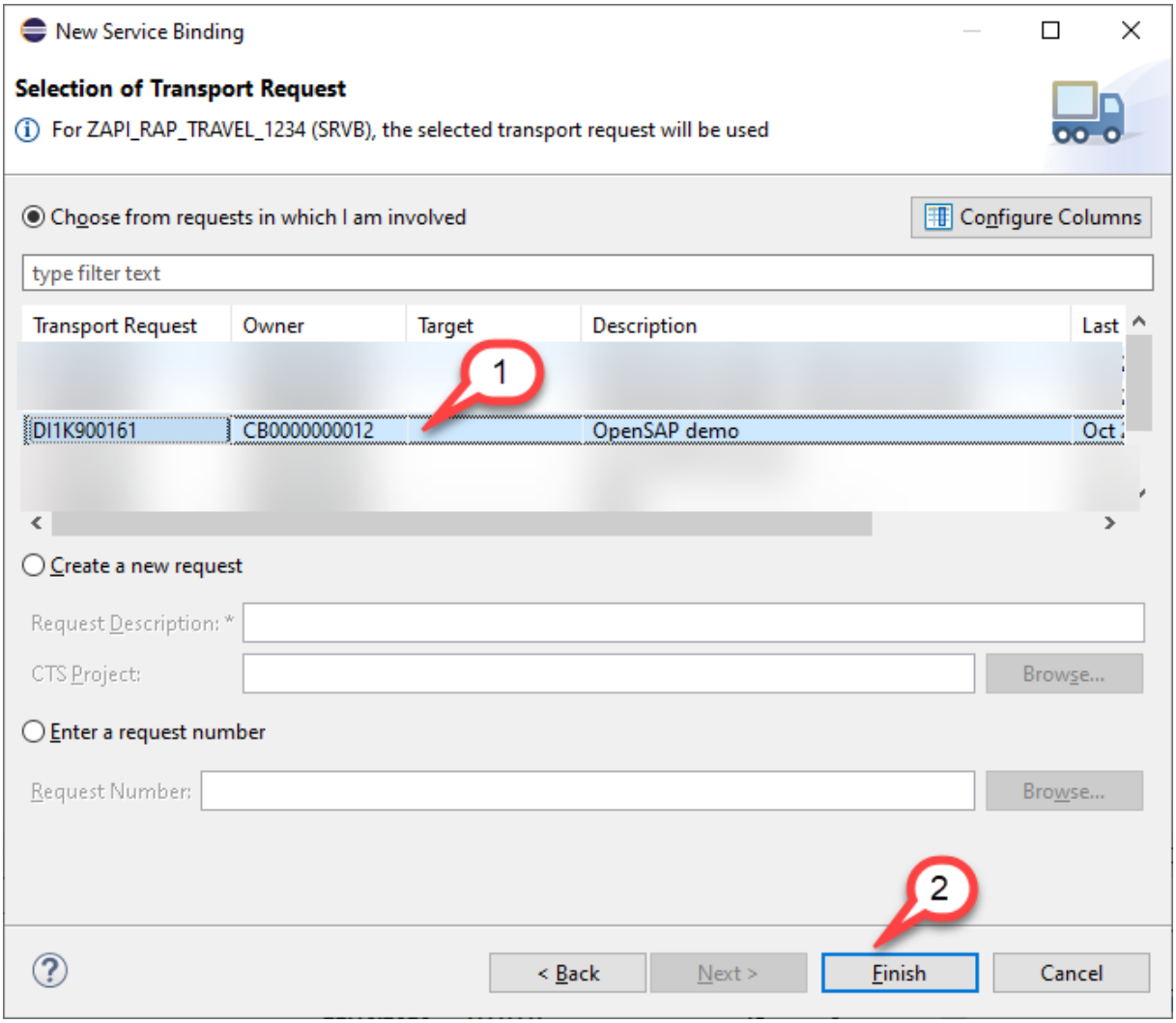


3. Entrez les valeurs suivantes
    - NomZAPI_RAP_TRAVEL_####
    - Description:API scenario
    - Type de reliure :OData V2 - Web API
    - Définition du service :ZAPI_RAP_TRAVEL_####

et appuyez sur **Suivant**

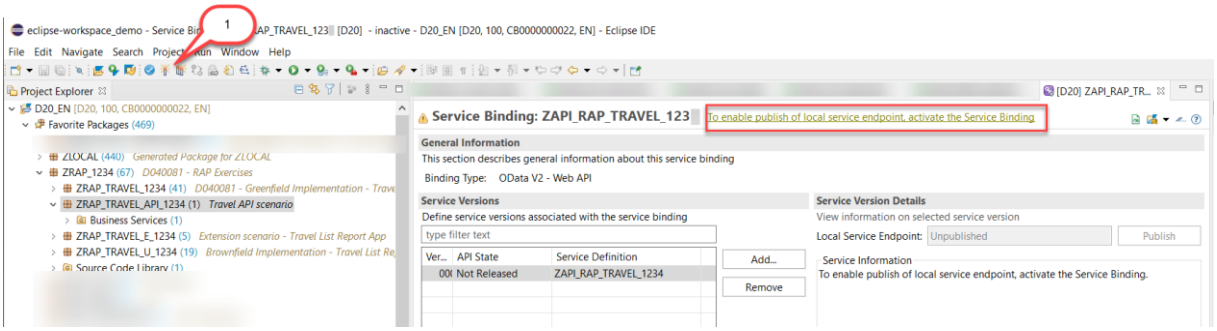


4. Sélectionnez une demande de transport et appuyez sur **Terminer**



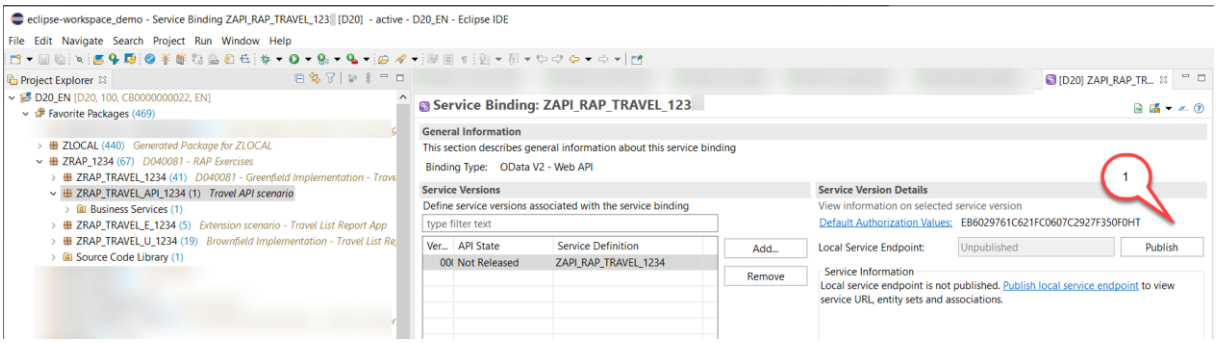
5. Activer la liaison de service

La liaison de service doit d'abord être activée.

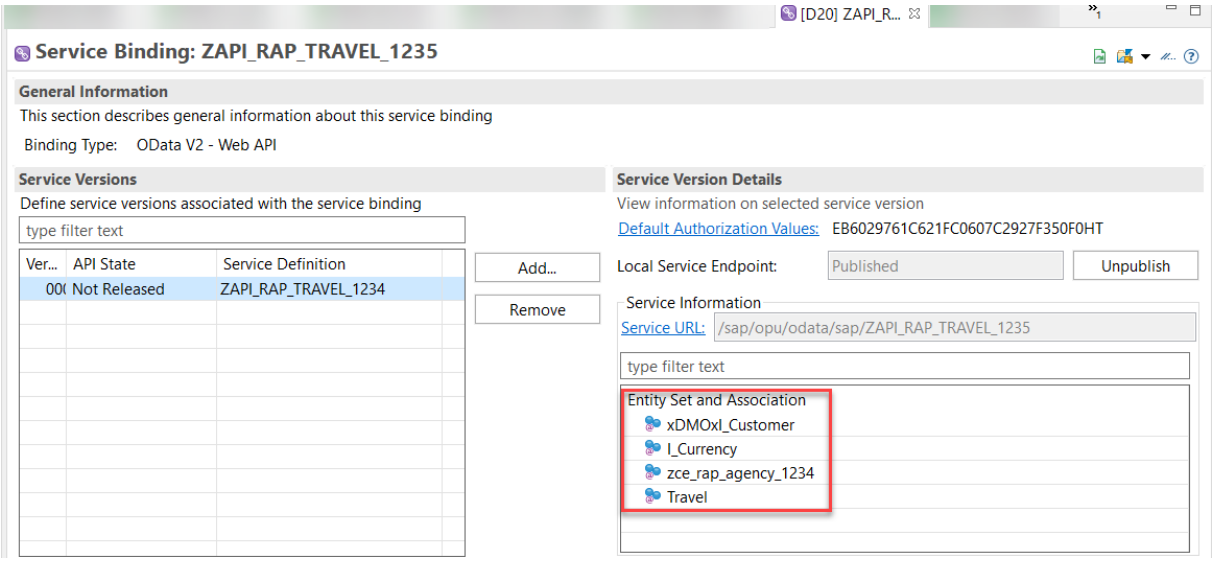


6. Publier la liaison de service

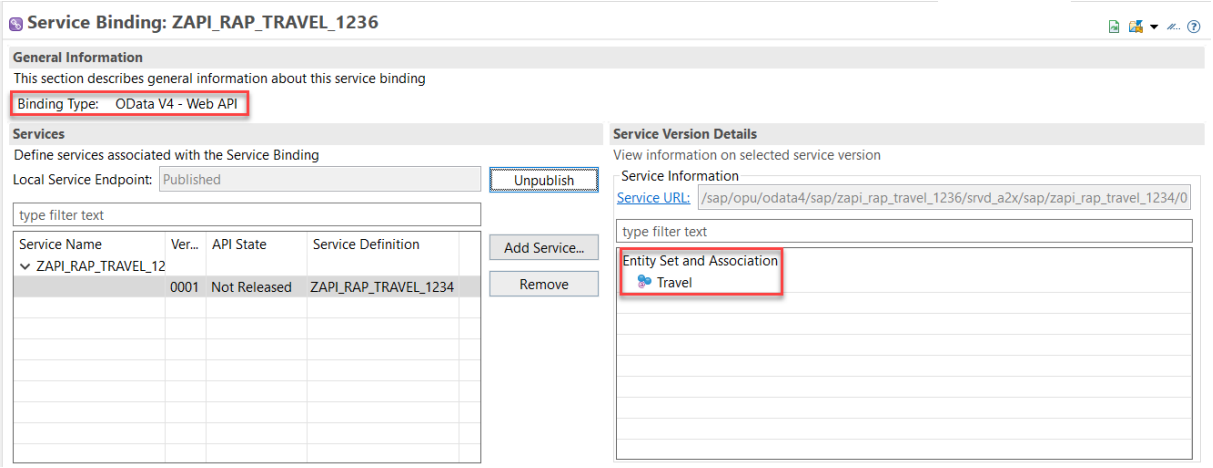
Une fois la liaison de service activée, vous pouvez appuyer sur le bouton **Publier** pour publier la liaison de service.



Veuillez noter que pour les API Web de type de liaison OData V2, les aides à la recherche sont également publiées automatiquement.



Ce comportement a changé avec OData V4 où le type de liaison OData V4 - Web API ne publie plus automatiquement les aides de valeur.



Résumé

Dans la première partie de cette unité, vous avez appris ce qui doit être fait pour publier une liaison de service pour une API Web.

Étant donné que la deuxième partie de la démo ne permet pas de créer un dispositif de communication sur un système d'essai, nous n'avons pas fourni ces étapes sous forme de script de guide pratique.