SAP CDS to Fiori

Create CDS Views and Use in Fiori App

Content:

Developing an App with SAP Fiori Elements using CDS view and Annotations (ABAP Programming Model) on SAP S/4 Hana.

I have created this blog using Multiple Blogs and combined them step by step to help you. I am starting to assume that you are too familiar with ABAP, Fiori, CDS Views, and Its tools.

So, Let's Start with ABAP Development Tools for CDS Views.

Prerequisites:

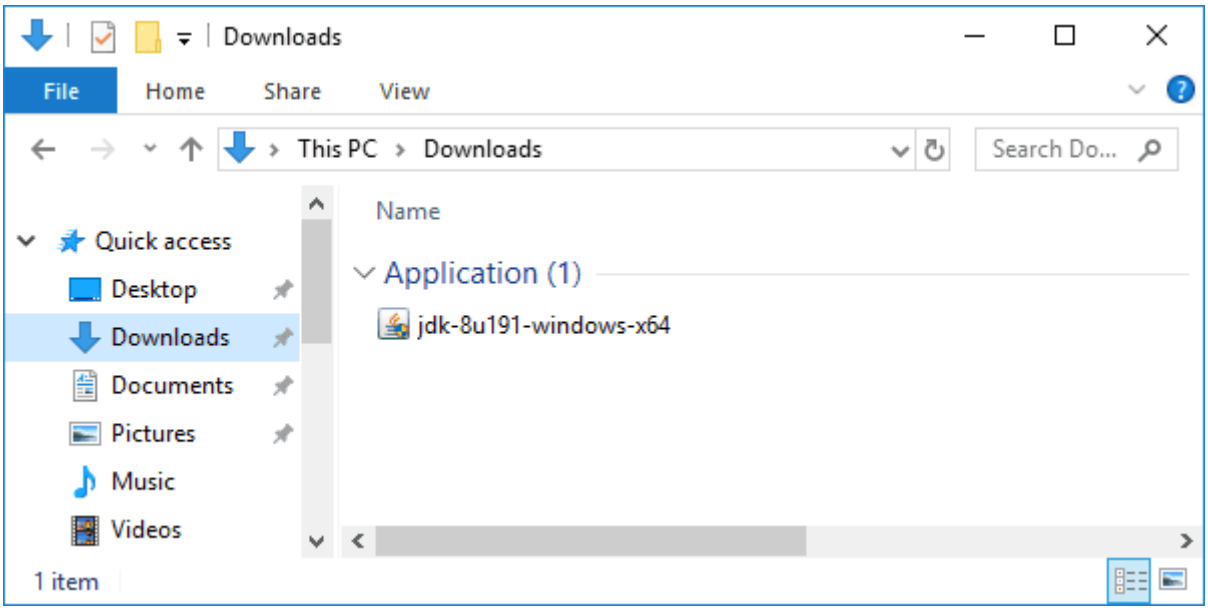
ADT is validated and tested against Java versions.

NOTE:

Please Skip This Installation If you already have Java Installed.

Install Java Version

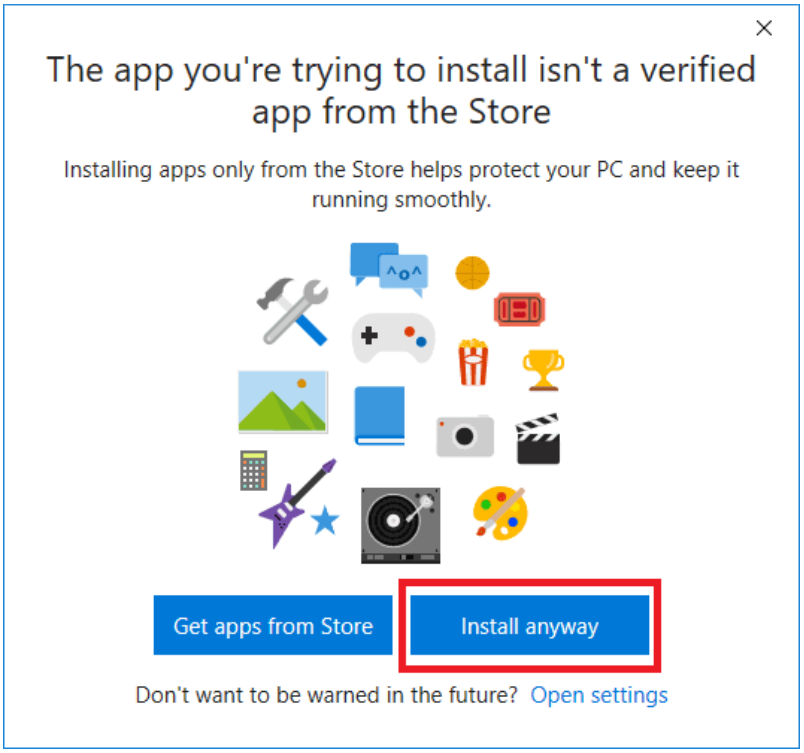
Open the location of the downloaded executable.



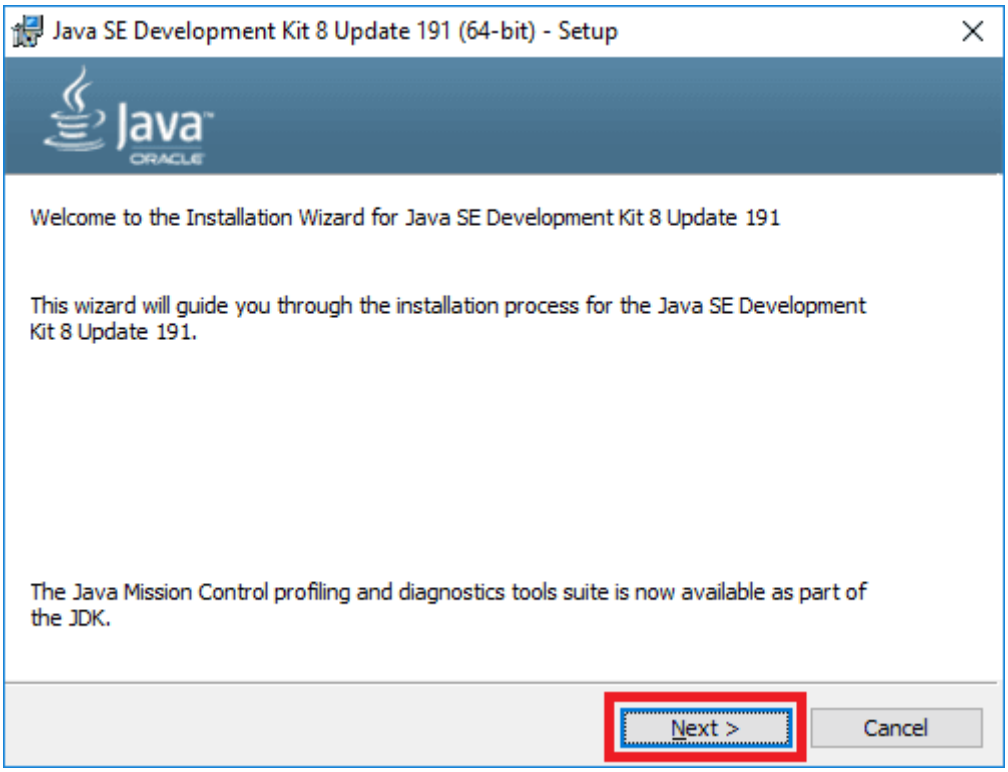
Double-click it to run the installer.

On Windows 10 a pop-up window will appear: The app you're trying to install isn't a verified app from the Store

Click on Install anyway.

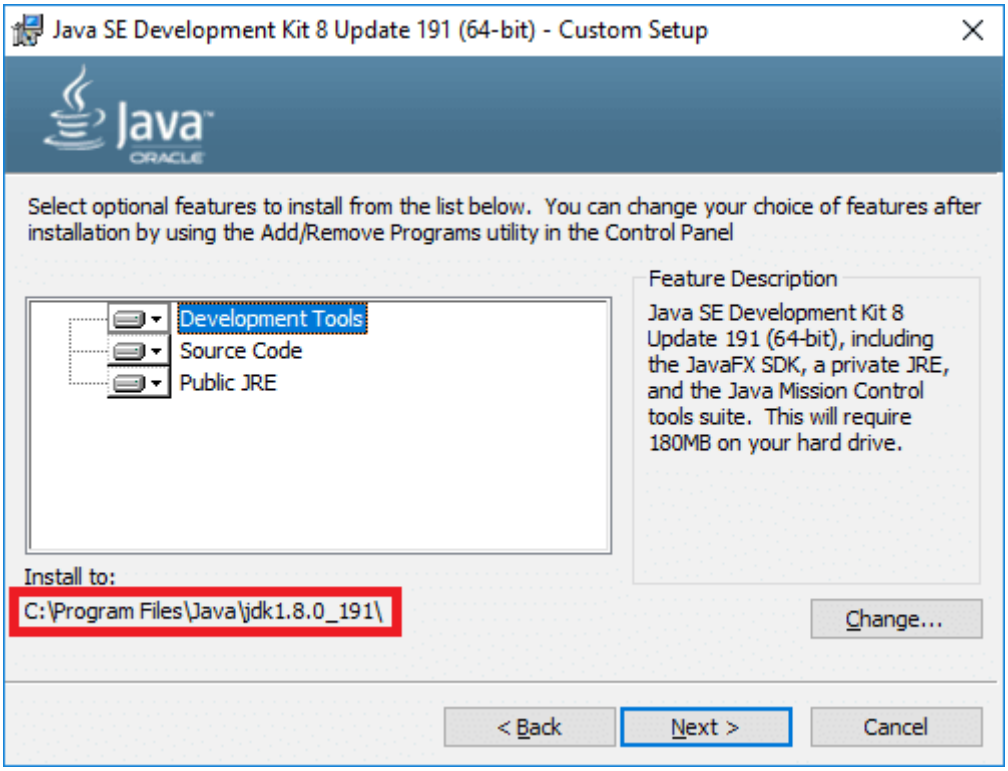


The JDK installer will start. Click Next.



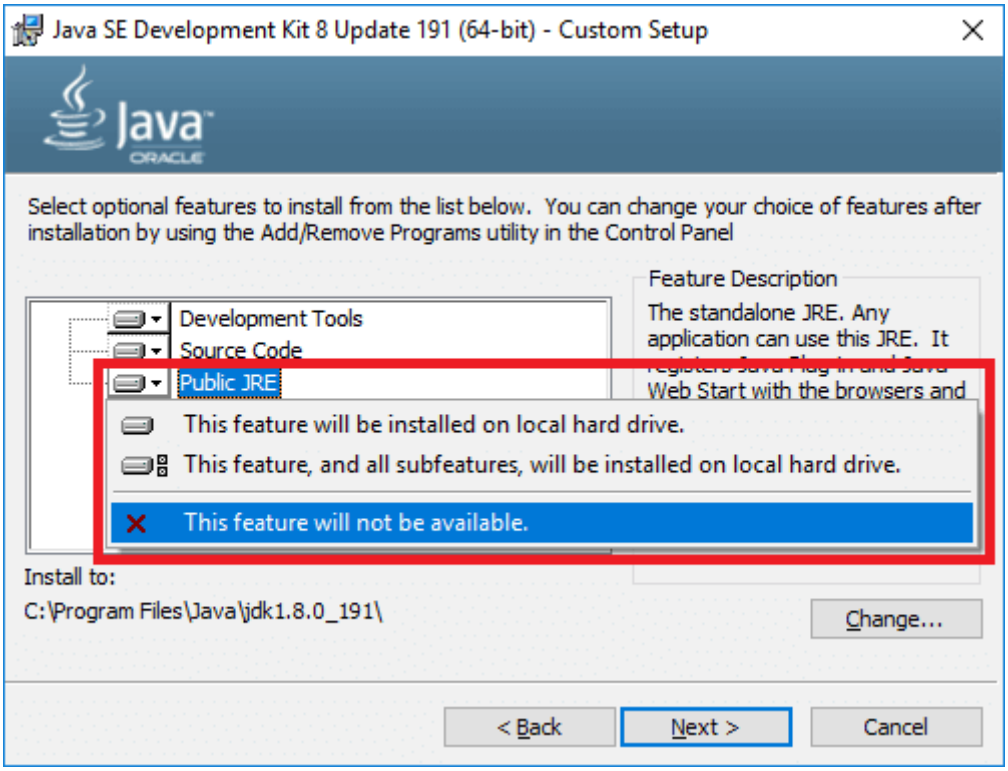
You can change the installation location by clicking on the Change… button.

In this example, we keep the default install location of C:\Program Files\Java\jdk1.8.0_191. From now on we will refer to this directory as [JAVA_INSTALL_DIR].

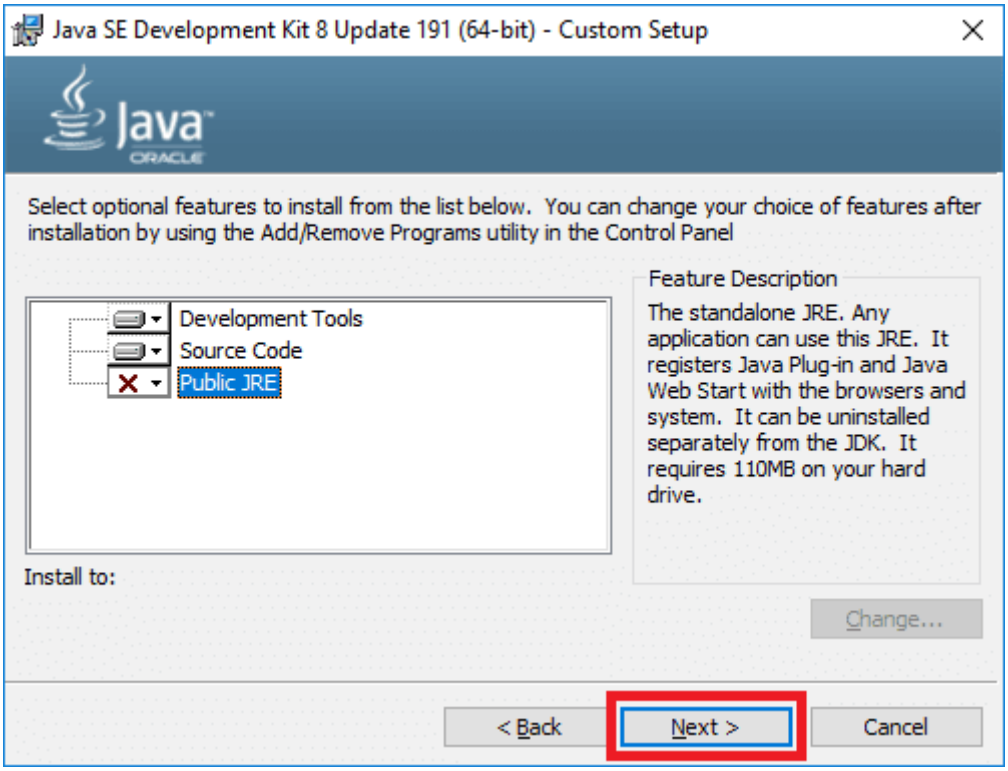


We will not install the public JRE as the JDK development tools already include a private JRE.

Select the Public JRE dropdown and click on This feature will not be available. as shown below.

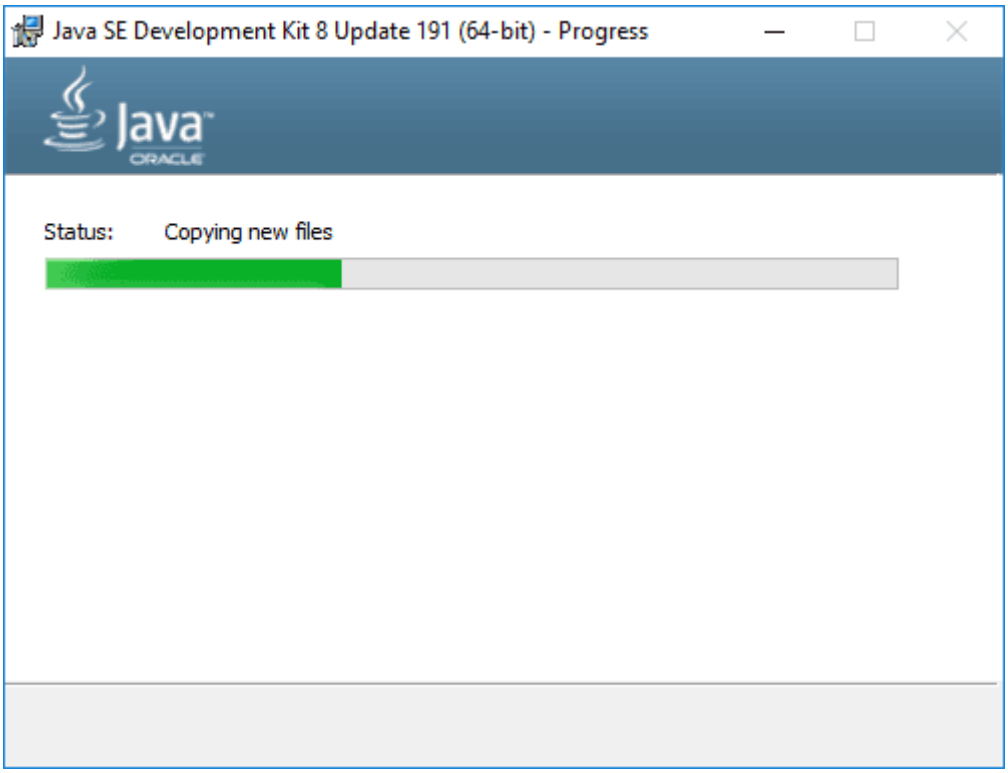


Click Next to start the installation.



The JDK installation will now start.

A progress bar shows the various steps that are executed.



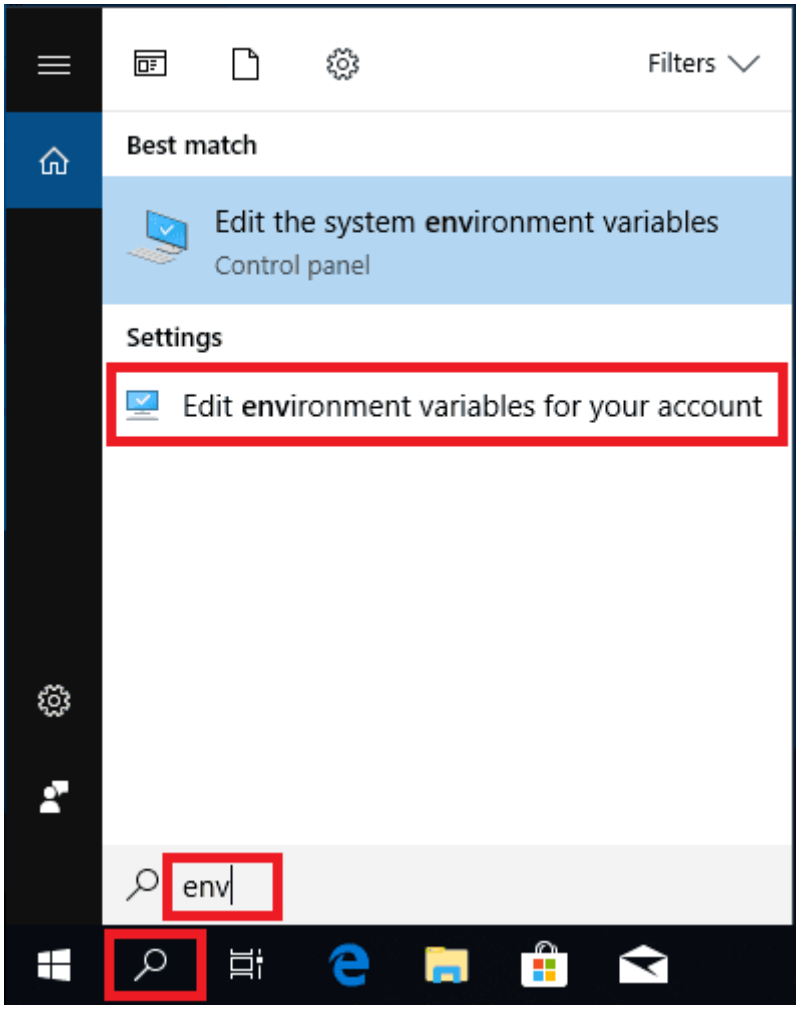
Once the installation is complete, click Close.

We need to set up an environment variable that will point to our JDK installation.

Click on the search button. Then type "env" (without quotes).

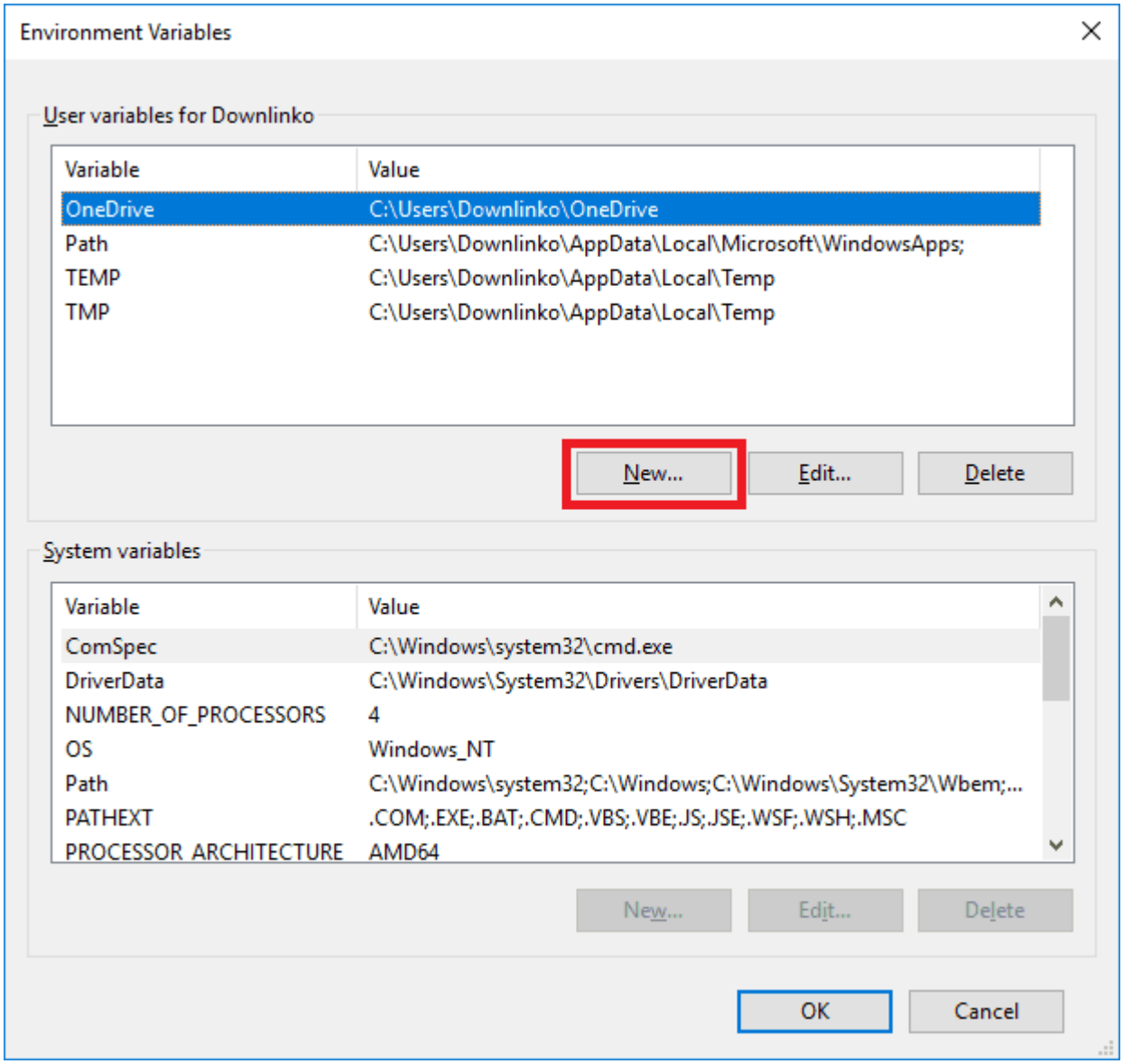
On Windows 7 click on the Windows button.

Click on the Edit environment variables for your account shortcut.



Wait for the environment variables window to open.

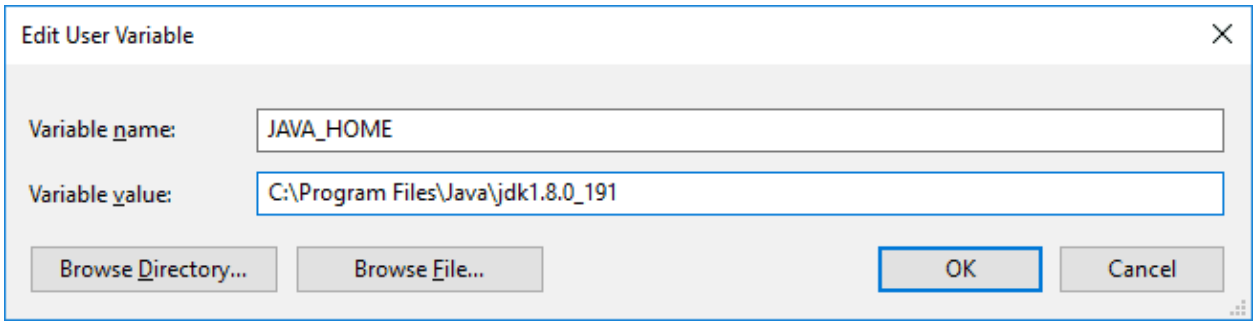
Click on New…



Enter "JAVA_HOME" as variable name. Enter the [JAVA_INSTALL_DIR] as variable value.

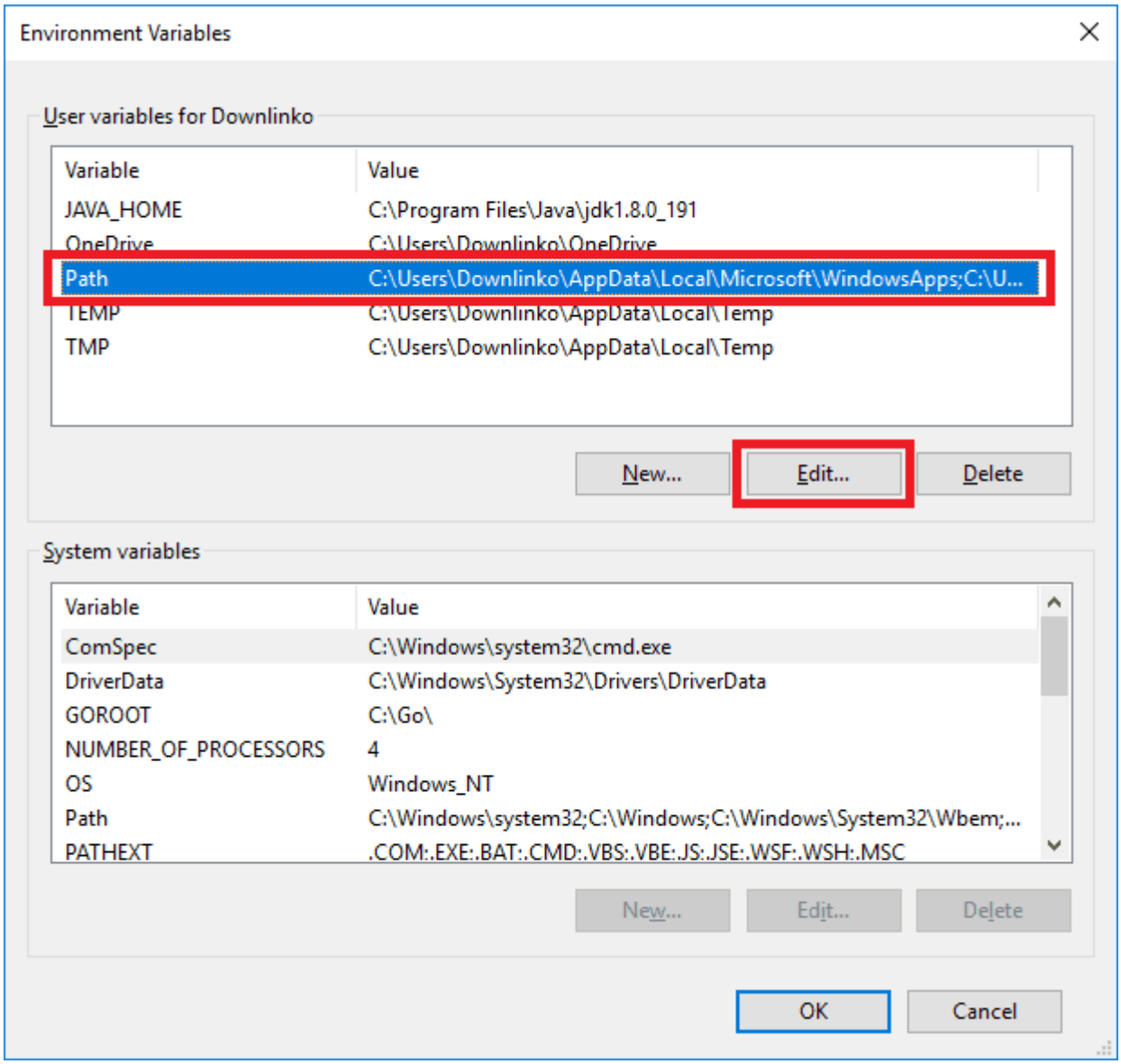
In this tutorial, the Java installation directory is C:\Program Files\Java\jdk1.8.0_191.

Click OK.



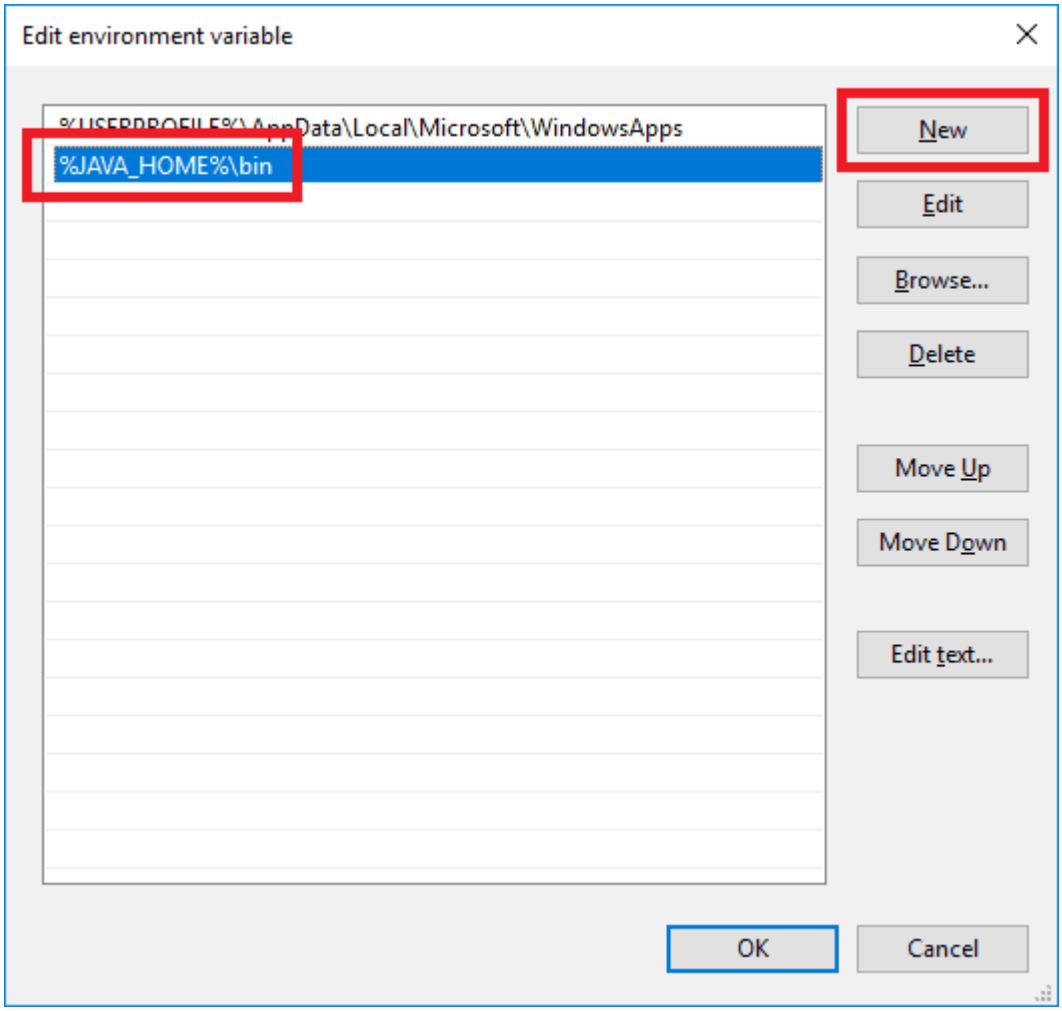
Next, we need to configure the PATH environment variable so we can run Java from a command prompt.

Select the Path variable. Click on Edit…

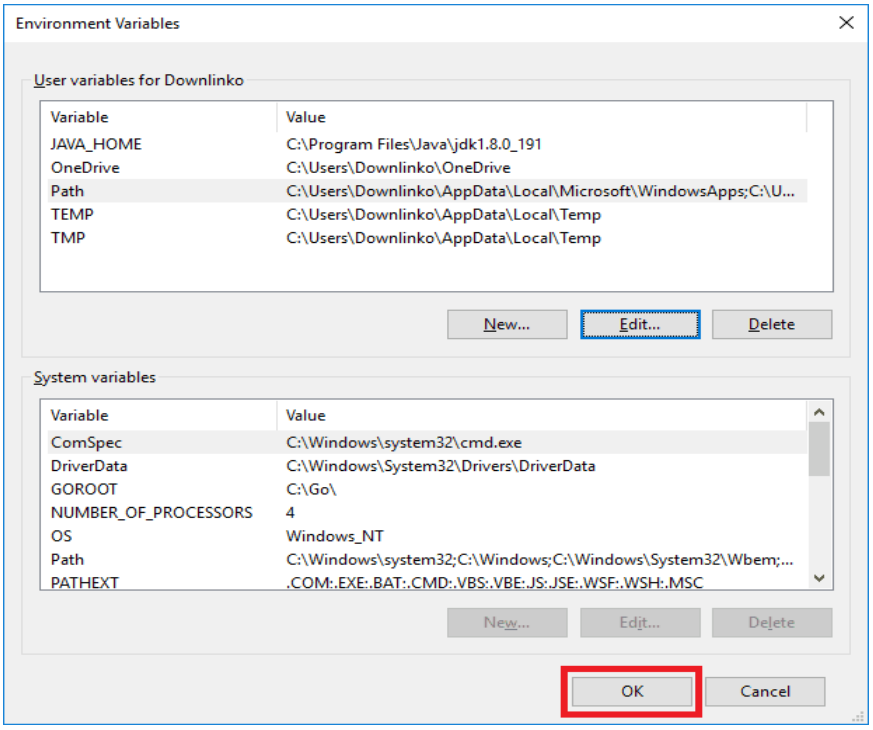


Click on New and type "%JAVA_HOME%\bin" as shown below.

Click OK.



Click OK once more to close the environment variables window.



If a Path *variable does not exist you need to create it. Use "Path" as variable name and "%JAVA_HOME%\bin" as variable value.*

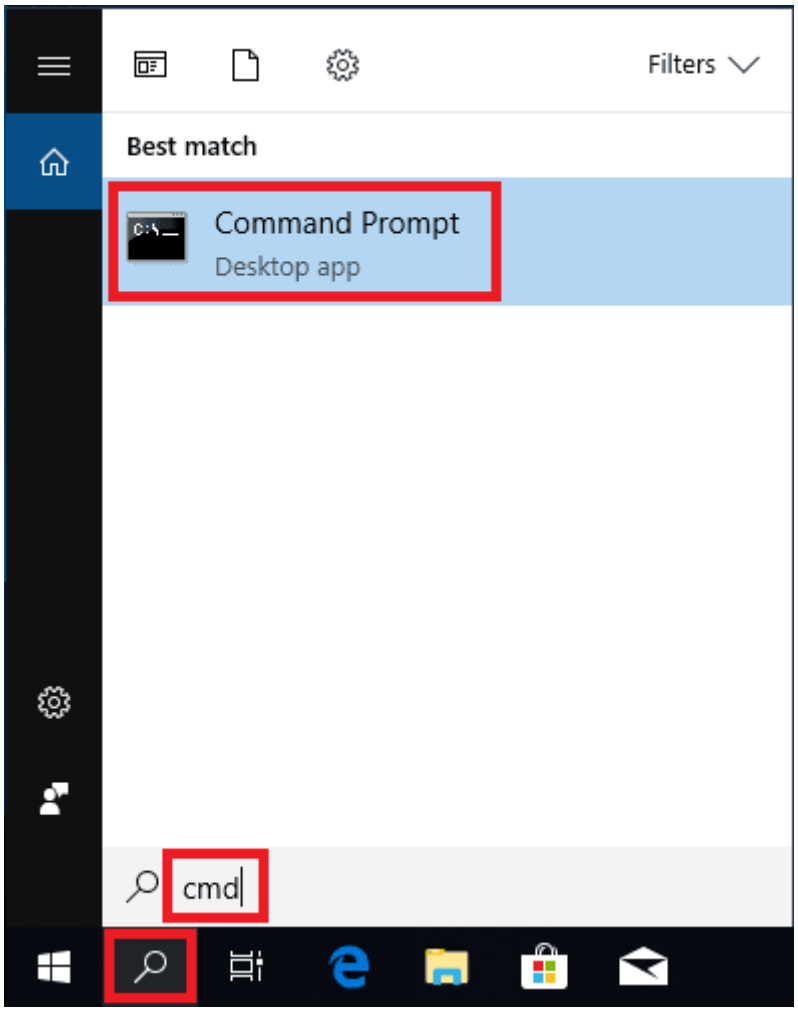
On Windows 7 you cannot add extra values for an existing Path *variable. You need to append ";%JAVA_HOME%\bin" at the end of the variable value instead.*

Test

Let's test the setup.

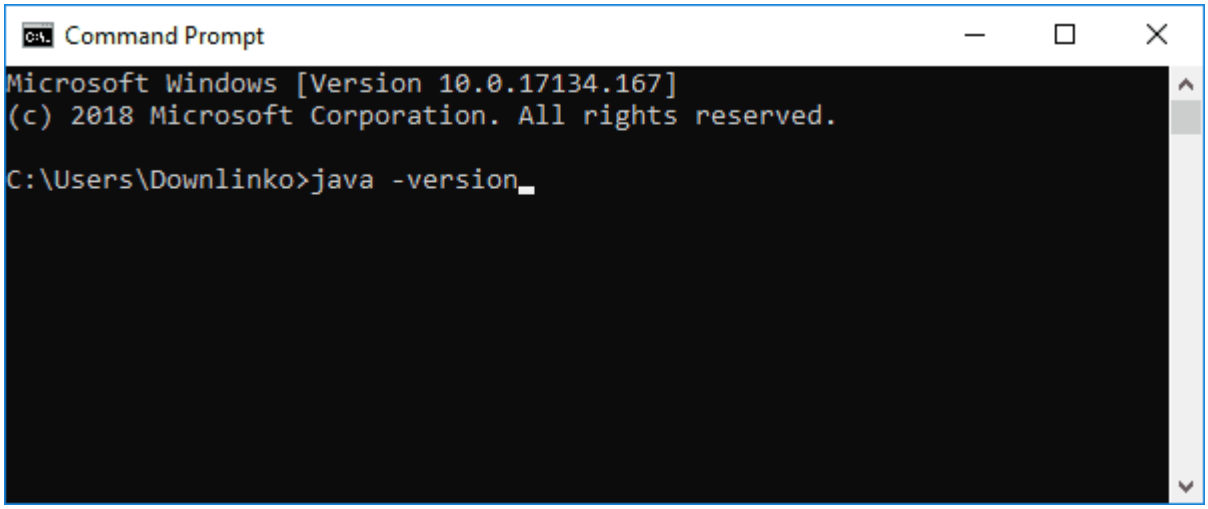
Click on the search button. Then type "cmd" (without quotes).

Click on the Command Prompt shortcut.

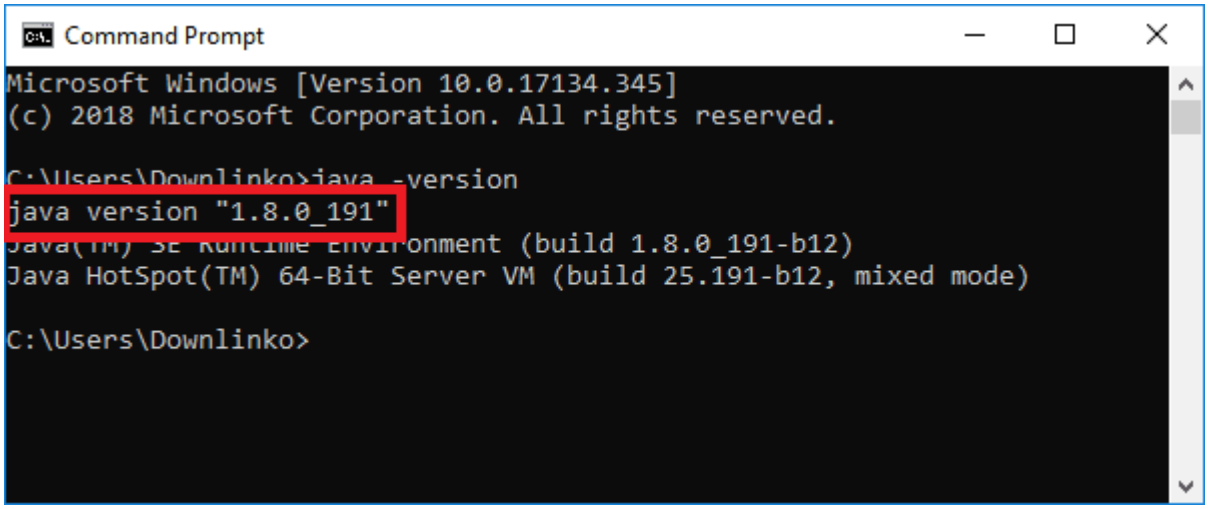


Wait for the command prompt to open.

Type "java -version" and press ENTER.



The above command prints the installed JDK version: 1.8.0_191.

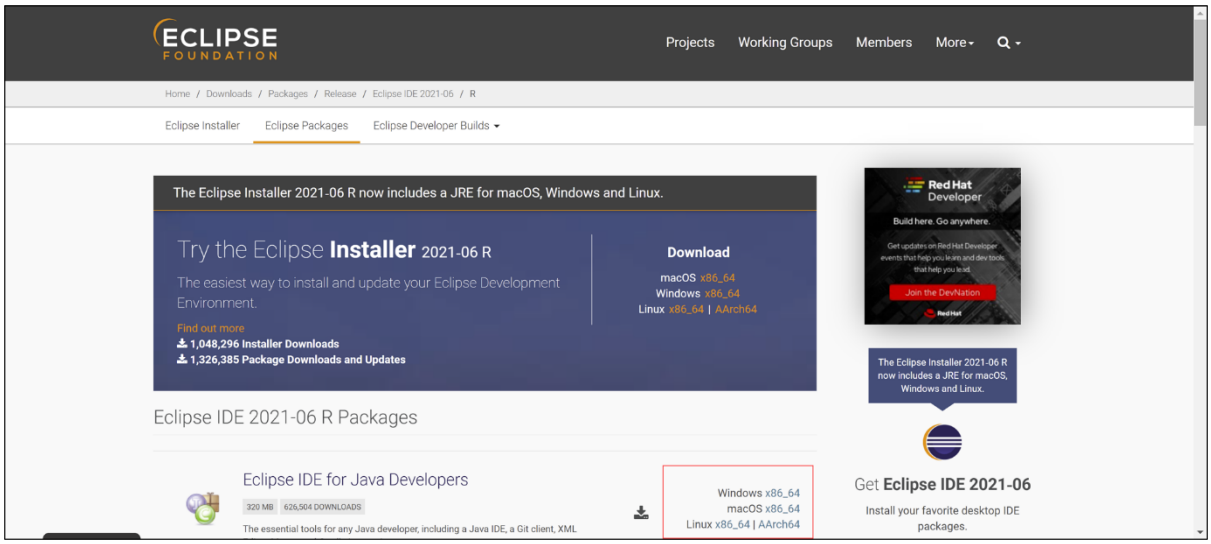


After completing the Prerequisites Start Installing Eclipse.

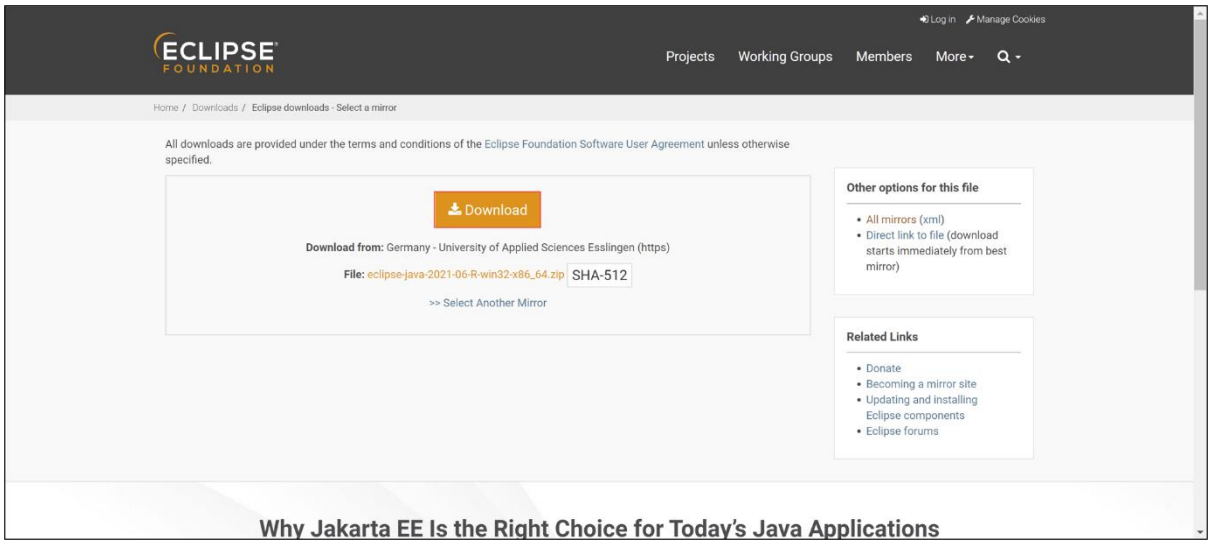
STEP 1

Install ABAP Development Tools (ADT)

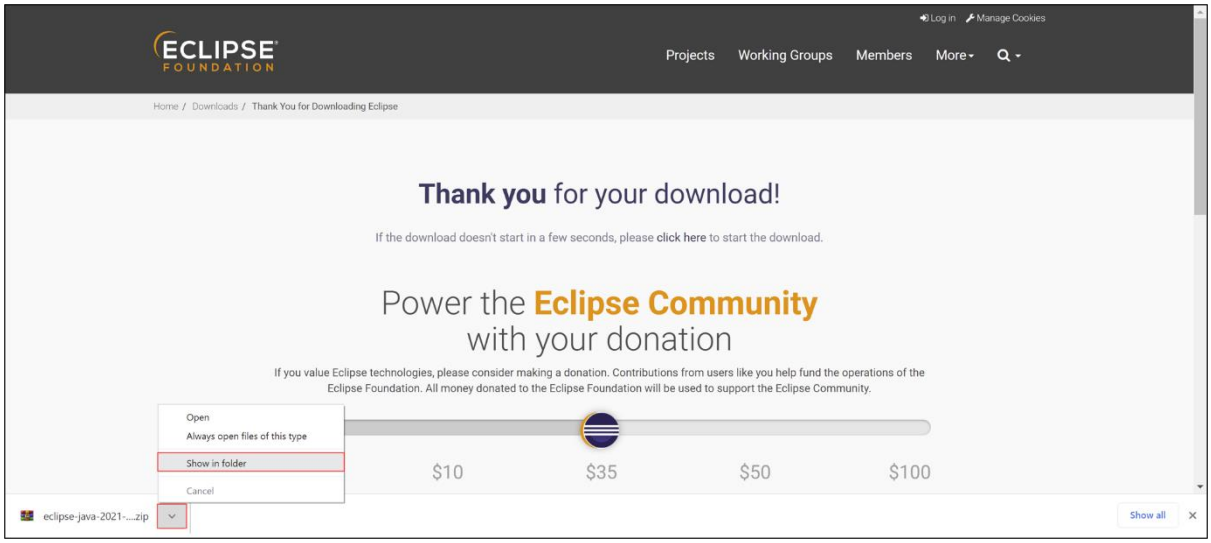
[1] Open the Eclipse download page to download the corresponding Eclipse version.



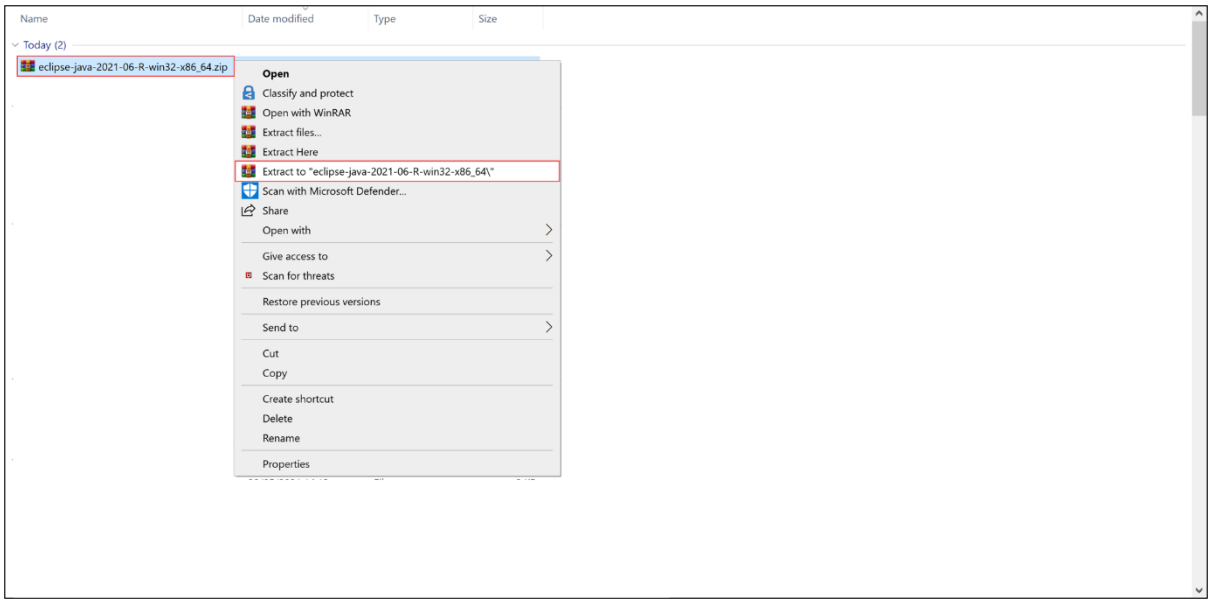
[2] Click Download.



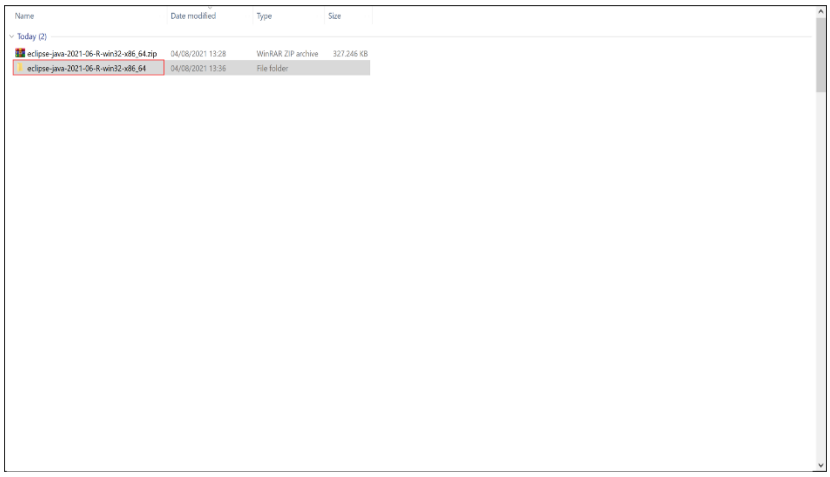
[3] Select Show in folder in your browser.



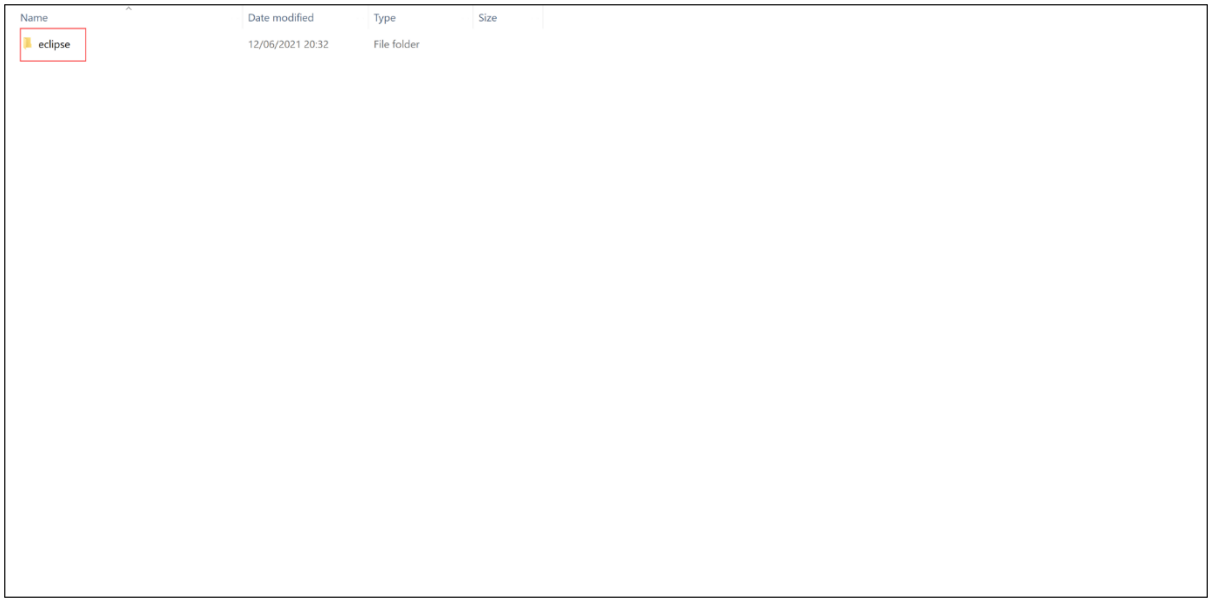
[4] Extract the Eclipse zip file with right-click.



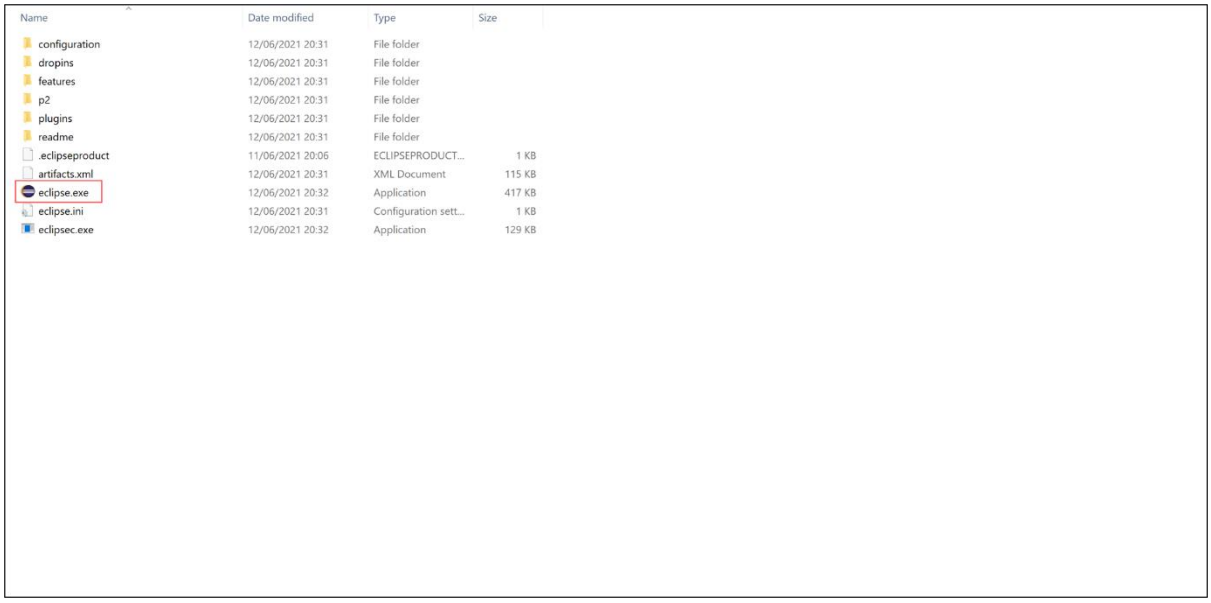
[5] Open the Eclipse-Java folder.



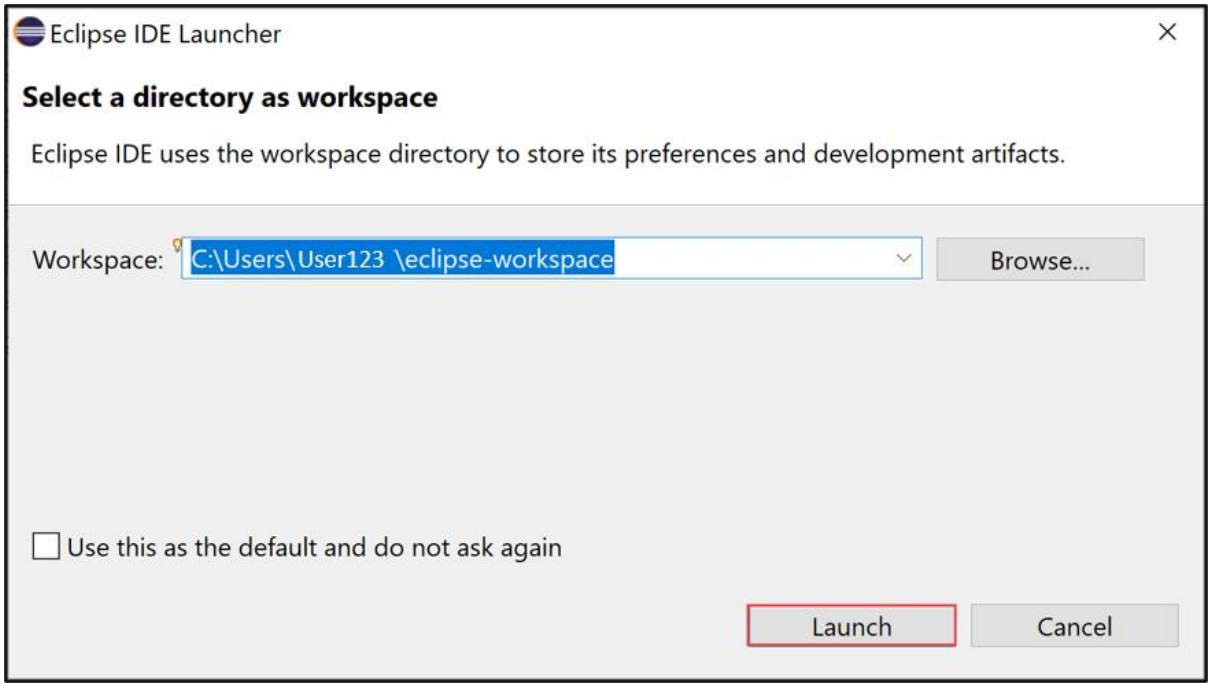
[6] Open the Eclipse folder.



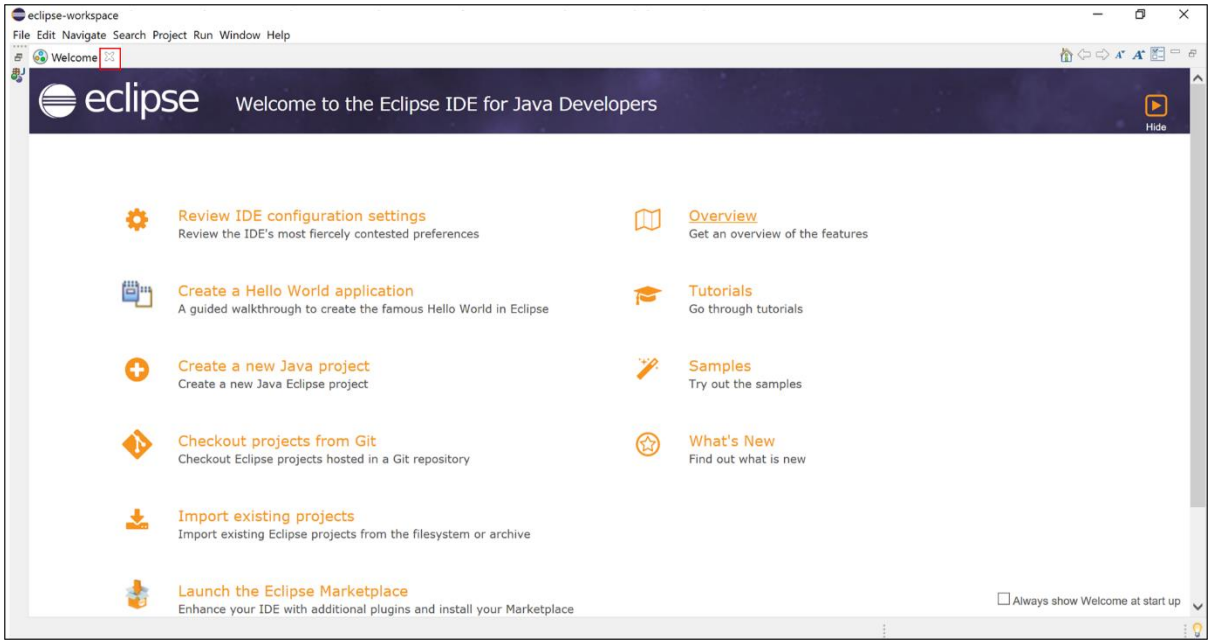
[7] Double-click eclipse.exe to run the application.

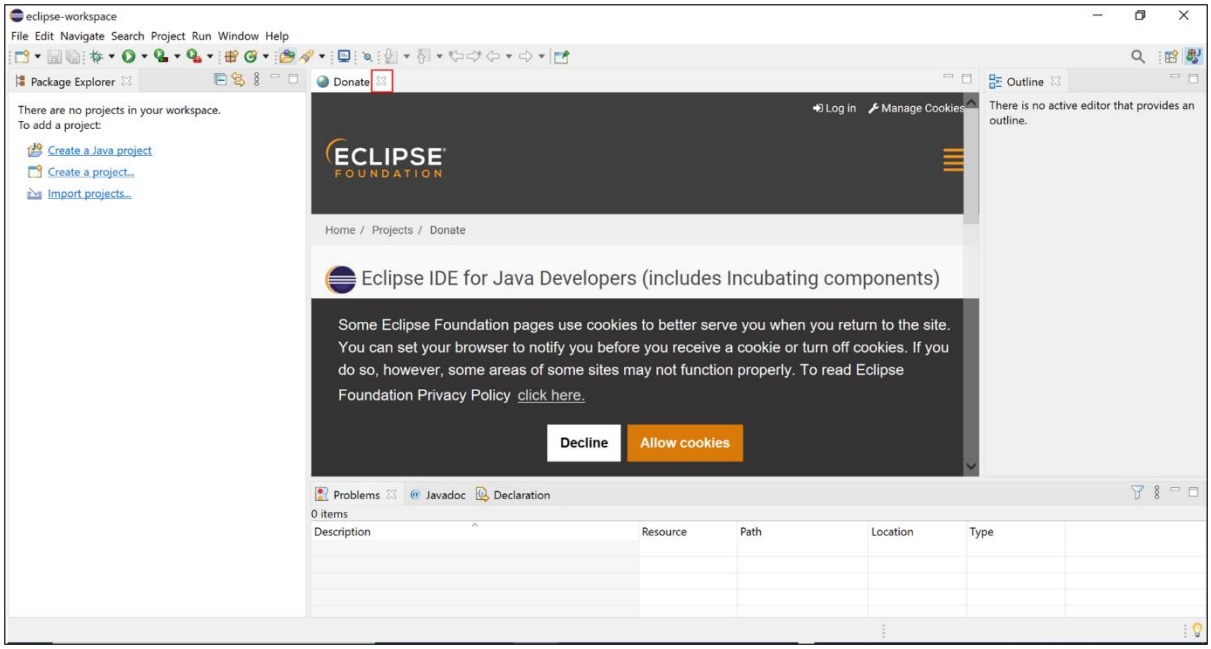


[8] Launch your workspace.

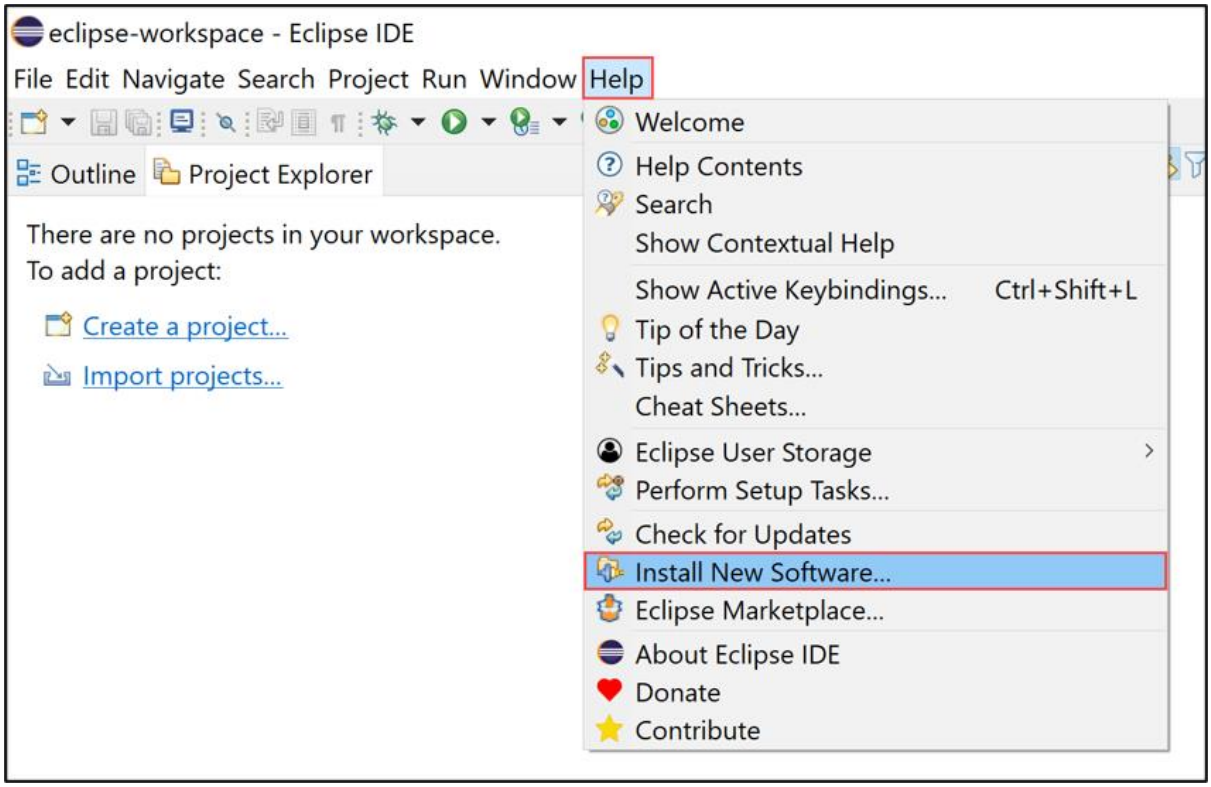


[9] Close both pages.

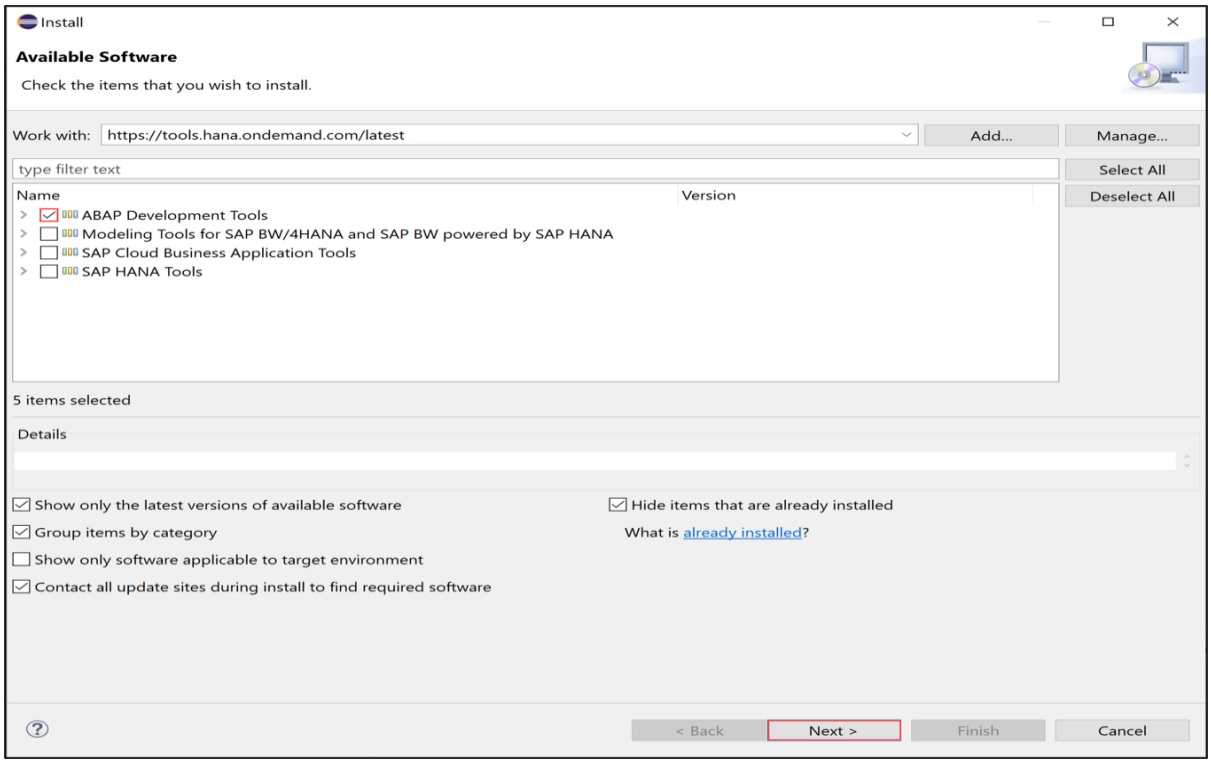




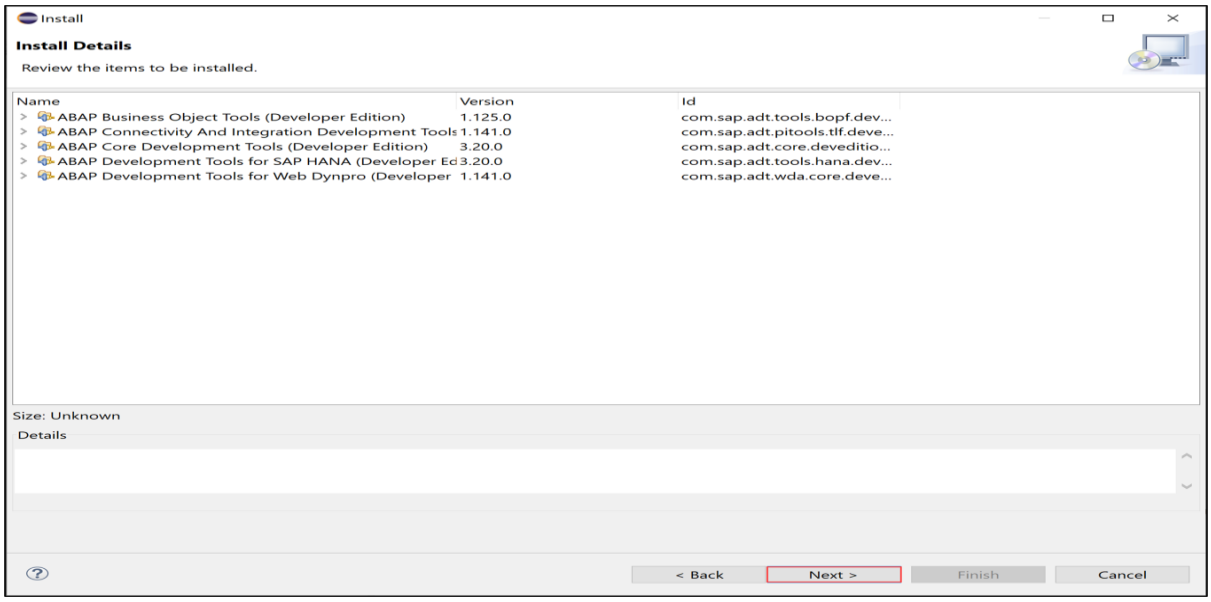
[10] Select Help > Install New Software.



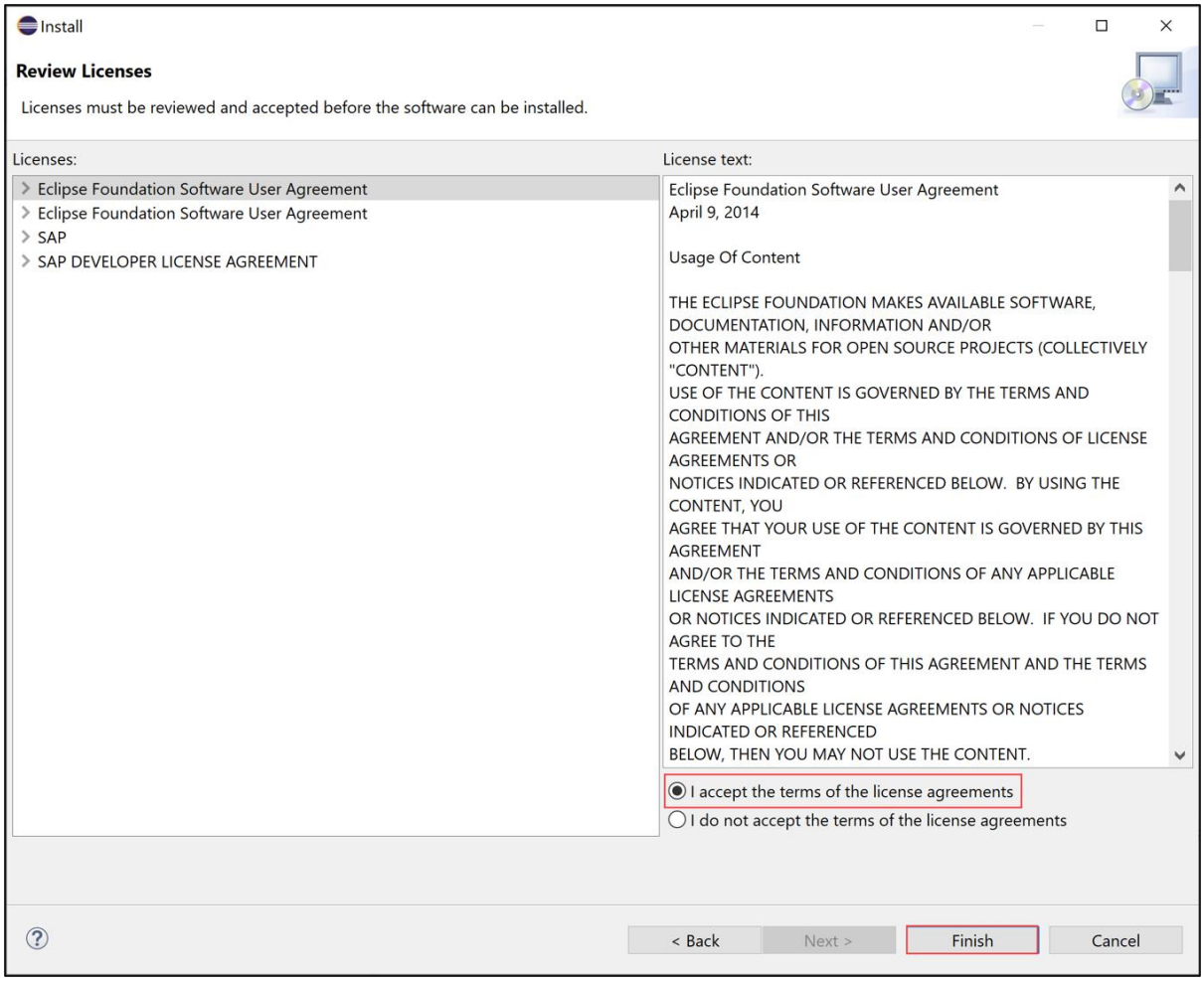
[11] Enter the latest ADT URL https://tools.hana.ondemand.com/latest in the Work with section, press enter, select ABAP Development Tools, and click Next >.



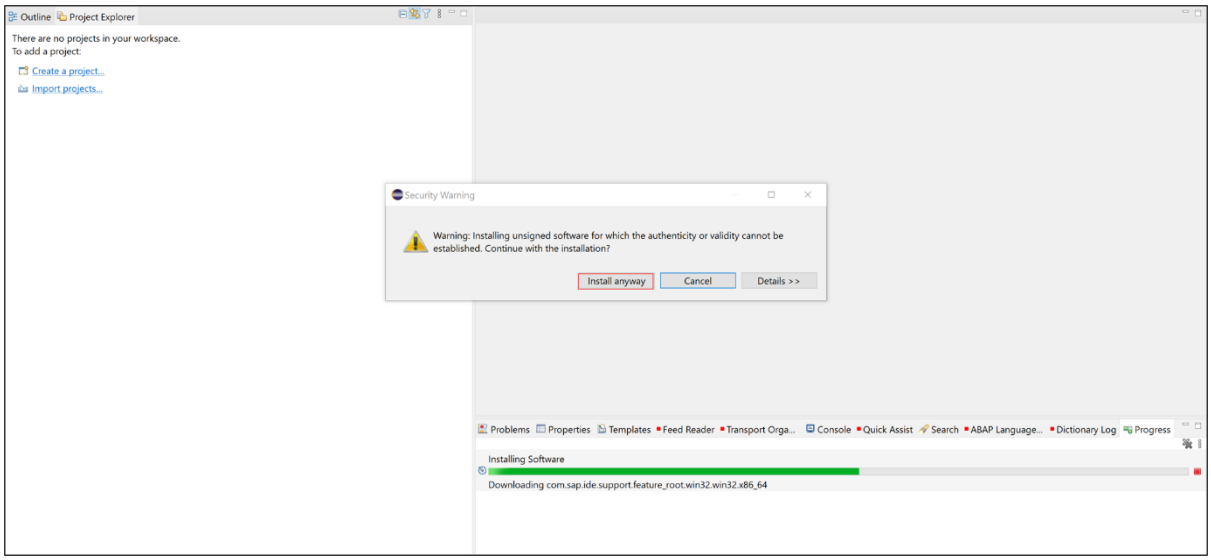
[12] Click Next >.



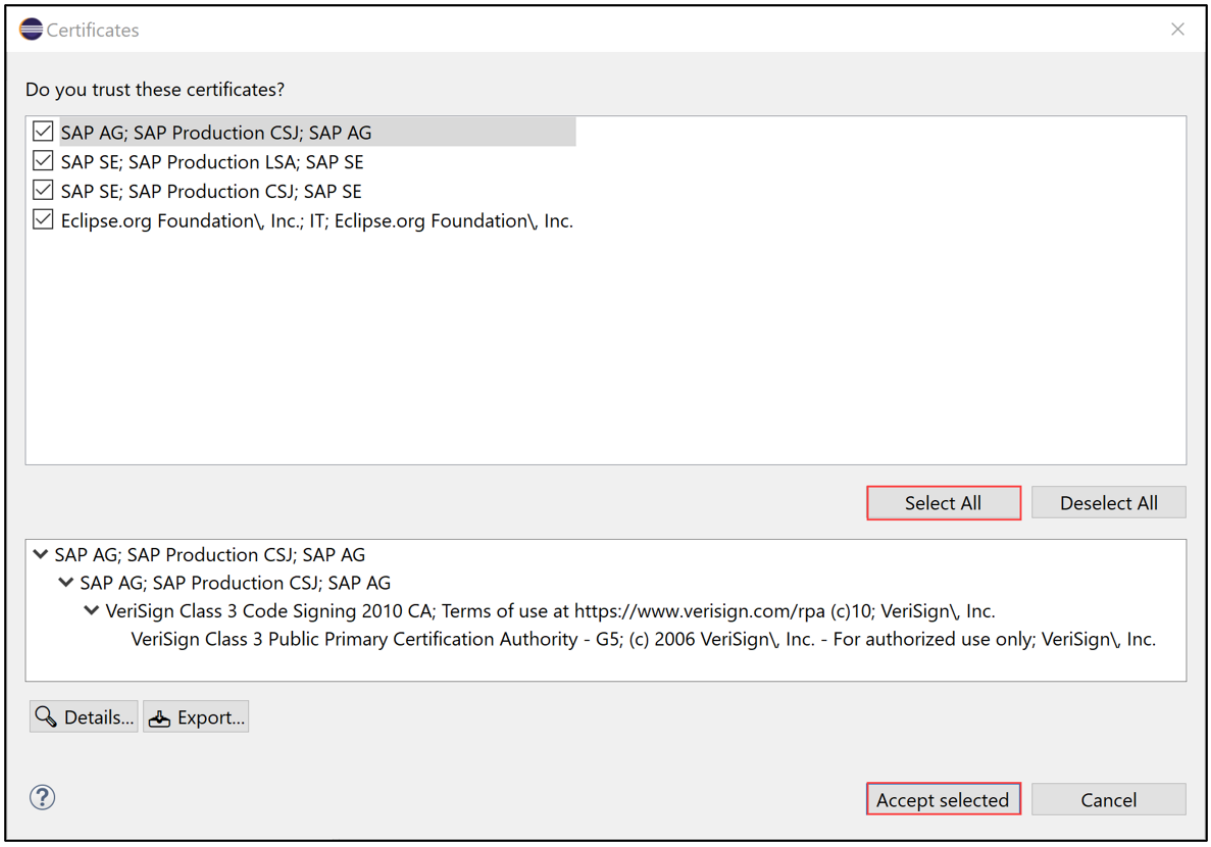
[13] Accept the license agreement and click Finish.



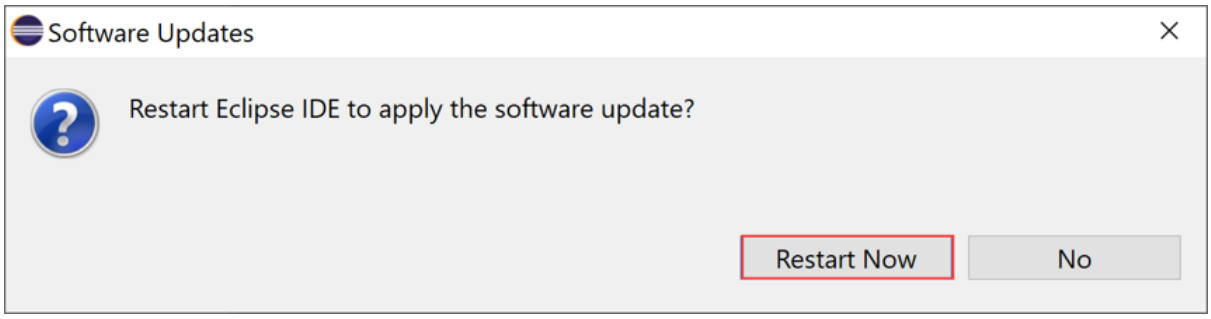
[14] Now ADT will be installed. Select Install anyway.



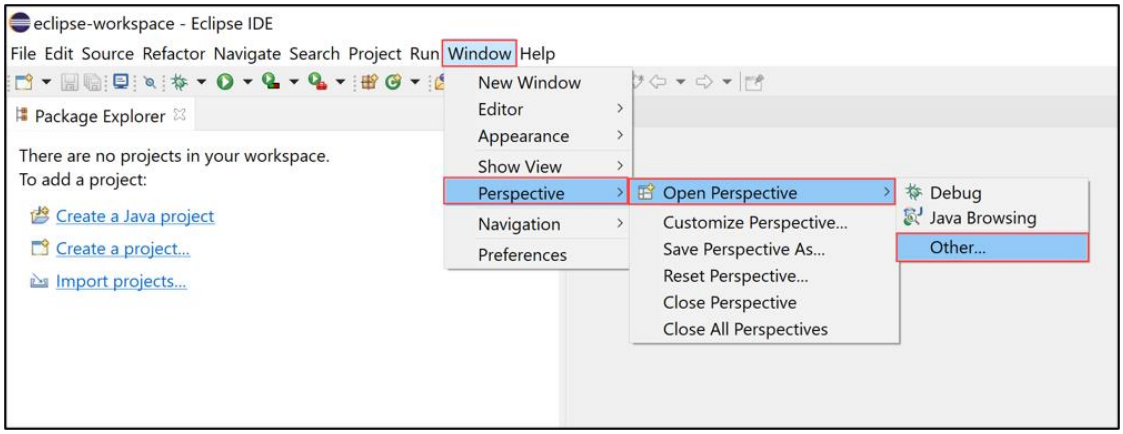
[15] Click Select All and Accept selected.



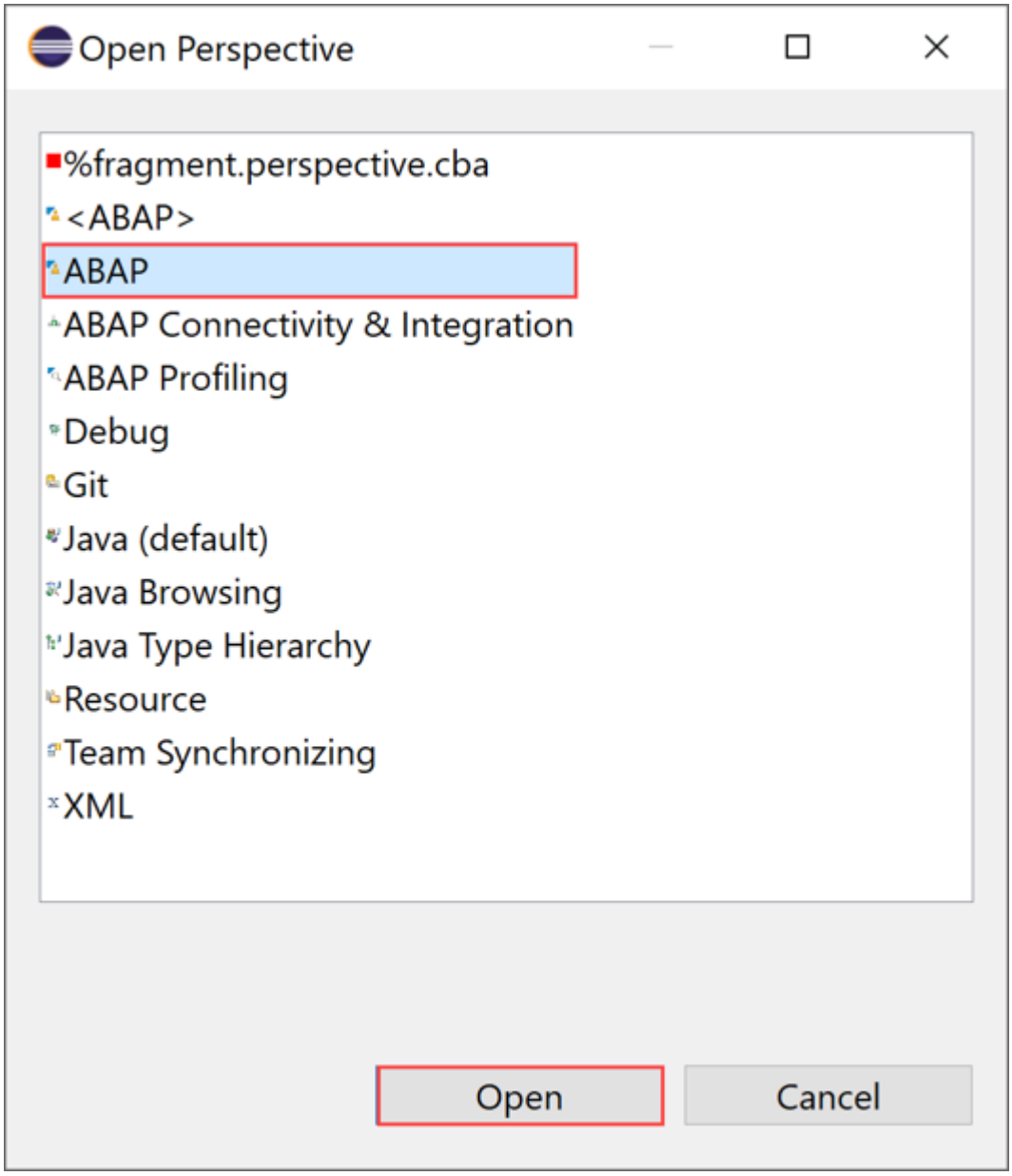
[16] Click Restart Now.



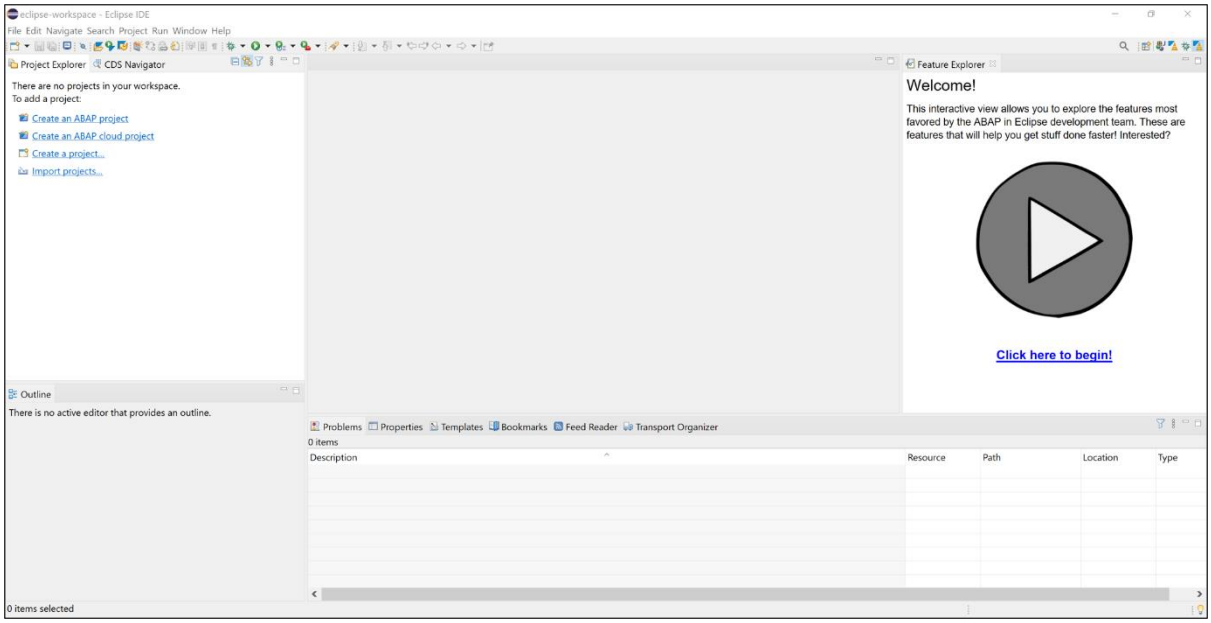
[17] Now ADT is installed. Switch to the ABAP perspective. Therefore select Window > Perspective > Other Perspective > Other.



[18] Then select ABAP and click Open.



[19] Check your result.



STEP 2

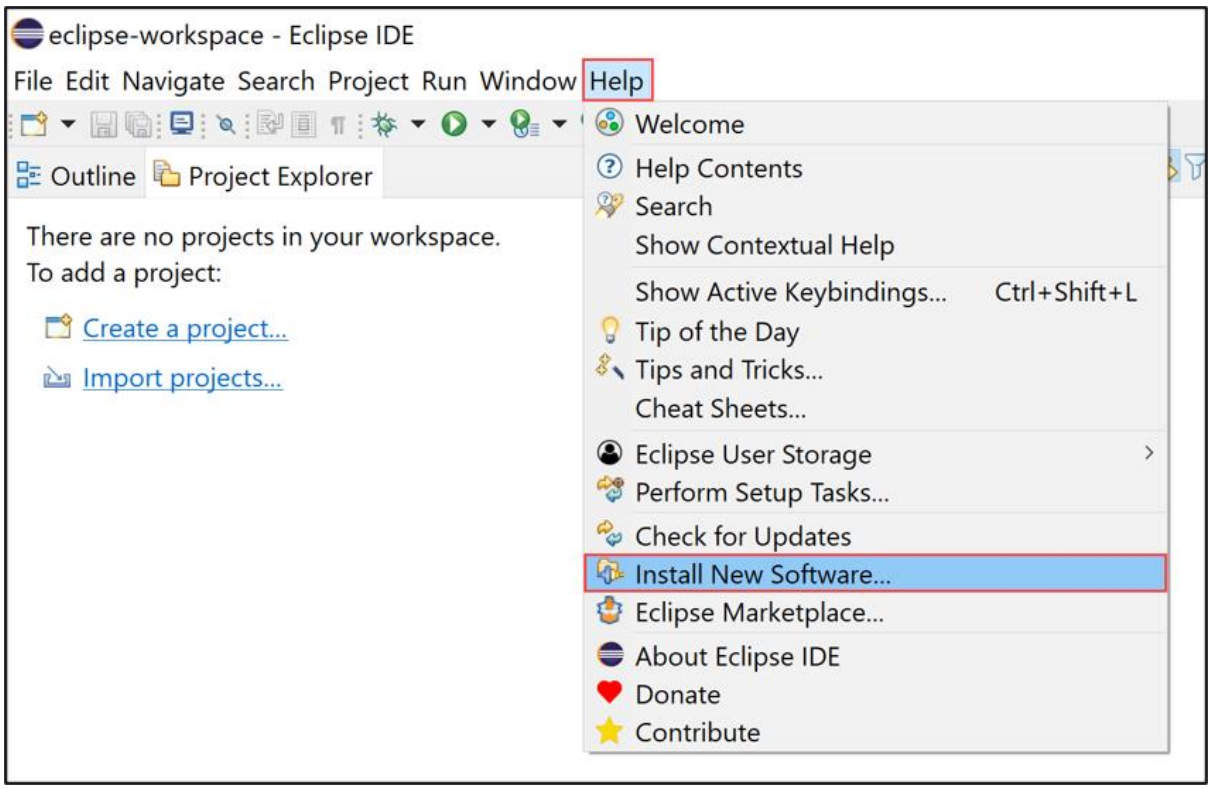
Install abapGit pluging:

HINT:

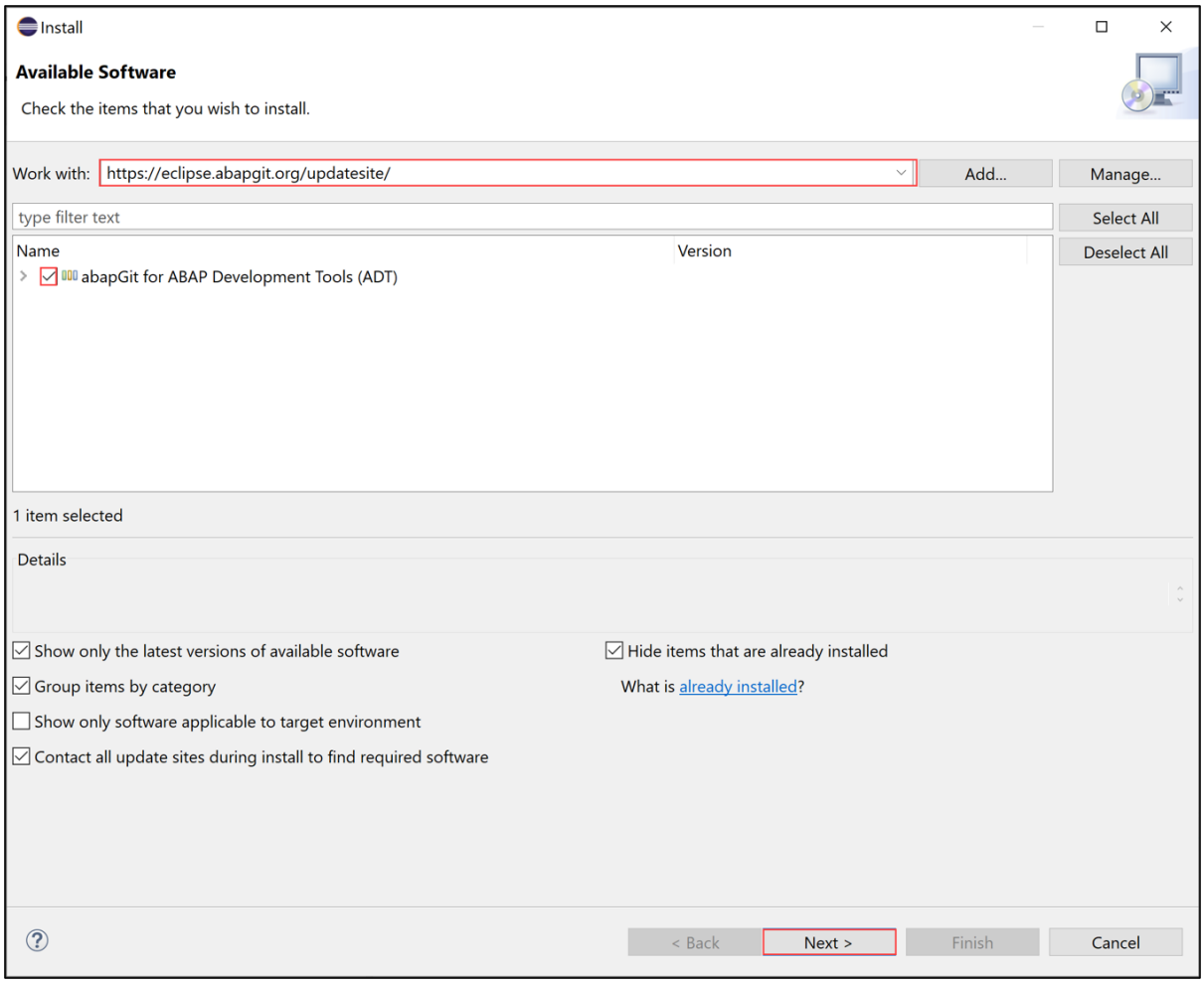
Step 2 is only mandatory for cloud users.

To transfer your ABAP development objects from on-premises SAP systems to an SAP BTP, ABAP Environment instance, you can use the abapGit plugin.

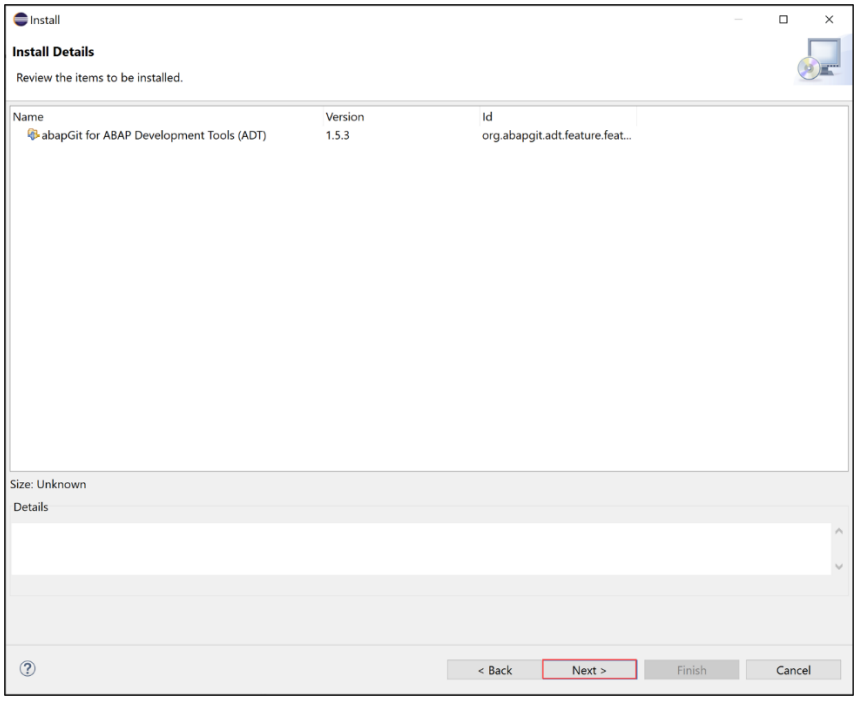
[1] Open Eclipse and select Help > Install New Software.



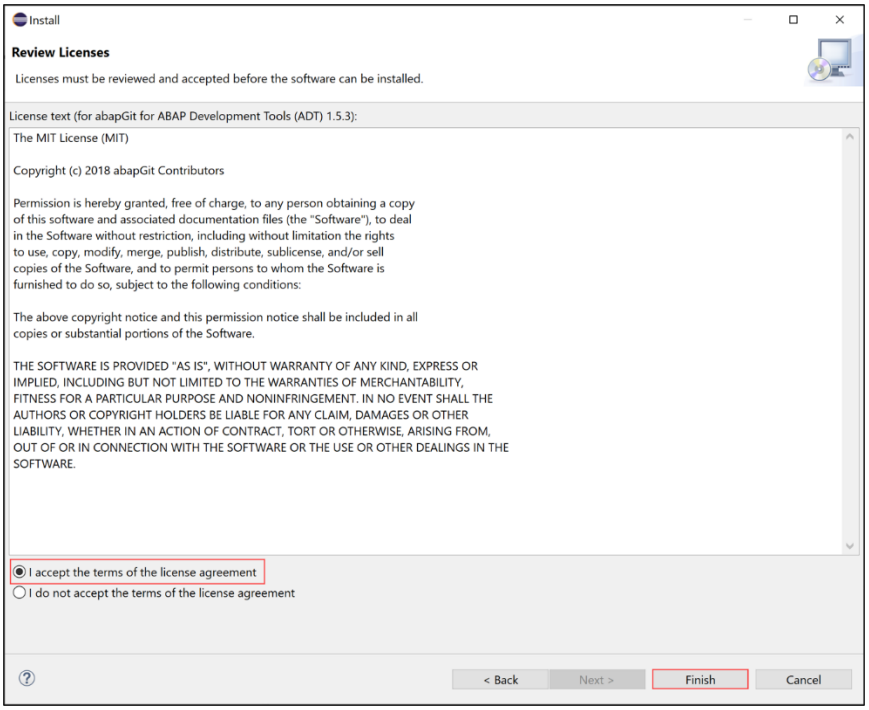
[2] Enter the abapGit URL https://eclipse.abapgit.org/updatesite/ in the Work with section, press enter, select abapGit for ABAP Development Tools (ADT) and click Next >.



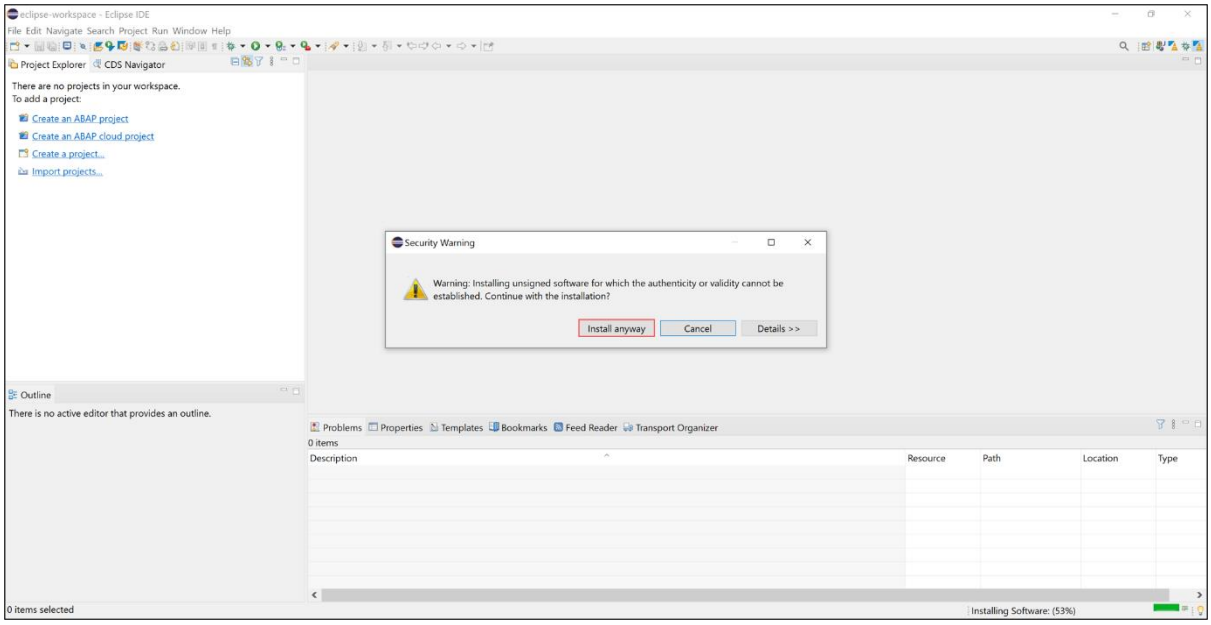
[3] Click Next >.



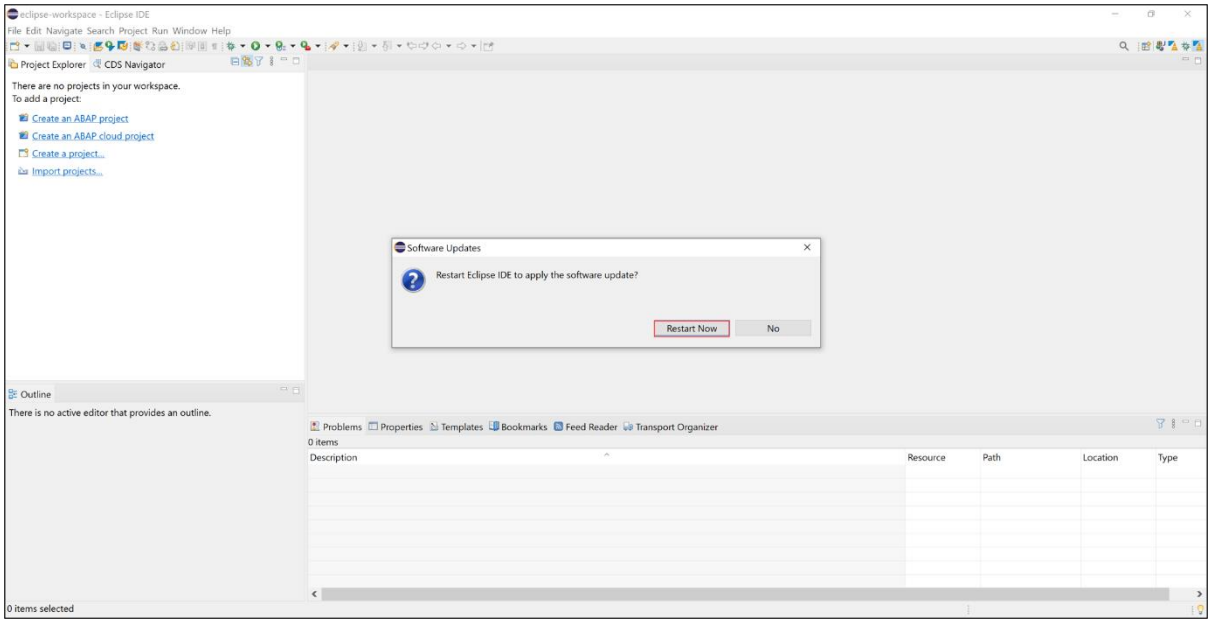
[4] Accept the license agreement and click Finish.



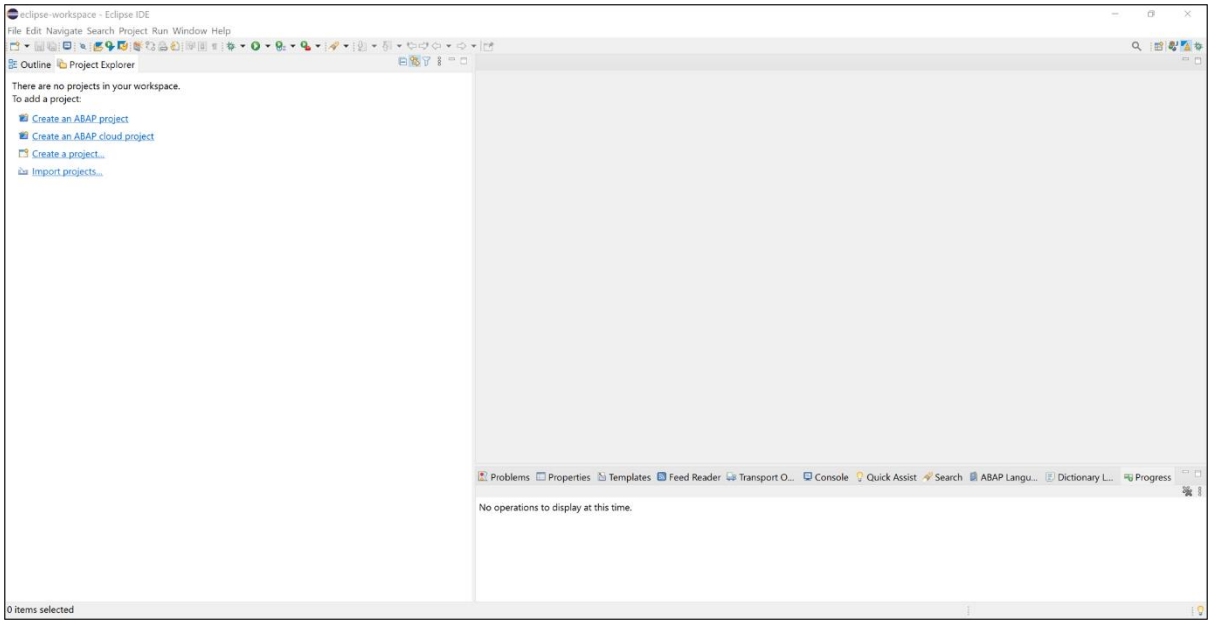
[5] Now ADT will be installed. Select Install anyway.



[6] Click Restart Now.



[7] Now abapGit for ADT is installed.



HINT:

Following this tutorial, you can update the latest version of Eclipse and ADT when new releases are available.

After The Successful Installation of ABAP Environment. We are going to Develop CDS View.

ABAP CDS View

To avoid spending time with table creations, we're going to reuse the Flight demo table offered by SAP, so let's create 2 new CDS views on top of this table:

ZDEMO_FLIGHT: Returns all the flights, type of plane, dates, and respective prices.

Code For ZDEMO_FLIGHT:

```
@AbapCatalog.sqlViewName: 'ZDEMOFLIGHT'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Flight'

@UI.headerInfo: {
   title.value: 'FlightCode',
   description.value: 'PlaneType',
   typeName: 'Flight',
   typeNamePlural: 'Flights'
}

@OData.publish: true

define view ZDEMO_FLIGHT
  as select from sflight
  association [0..*] to ZDEMO_FLIGHT_CHART as _Chart on  $projection.FlightCode =
_Chart.FlightCode
                                     and $projection.FlightDate = _Chart.FlightDate
{
    @EndUserText.label: 'Flight Code'
    @UI: {
       lineItem.position: 10,
       fieldGroup: {
         qualifier: 'FlightDetails',
         position: 10
       }
    }
  key concat(carrid, connid) as FlightCode,

    @UI: {
       selectionField.position: 10,
       lineItem.position: 20,
       fieldGroup: {
         qualifier: 'FlightDetails',
         position: 20
       }
    }
  key fldate               as FlightDate,
```

```
    @UI.lineItem.position: 30
    @Semantics.amount.currencyCode: 'Currency'
    price               as Price,

    @Semantics.currencyCode: true
    currency            as Currency,

    planetype           as PlaneType,

    _Chart
}
```

ZDEMO_FLIGHT_CHART: Returns the maximum capacity of seats and occupied seats per class for each one of the flights.

Code for ZDEMO_FLIGHT_CHART:

```
@AbapCatalog.sqlViewName: 'ZDEMOFLIGHTCHART'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Flight'

@UI.chart: [{
   qualifier: 'OccupiedSeats',
   chartType: #COLUMN,
   dimensions:  [ 'FlightCode' ],
   measures:  [ 'MaximumCapacity', 'EconomyOccupiedSeats', 'BusinessOccupiedSeats',
'FirstOccupiedSeats' ]
}]

define view ZDEMO_FLIGHT_CHART
  as select from sflight
{
  key concat(carrid, connid) as FlightCode,

  key fldate            as FlightDate,

    seatsmax            as MaximumCapacity,

    seatsocc            as EconomyOccupiedSeats,

    seatsocc_b          as BusinessOccupiedSeats,

    seatsocc_f          as FirstOccupiedSeats
}
```

Let's review some important points about the annotations we have in these CDS views:

- @UI.headerInfo: This annotation is used to place information in the header of the Object Page, in our case we place the Flight Code and Plane Type as title and description.
- @UI.lineItem: This annotation determines the position of the field in the result list of the List Report.
- @UI.selectionField: This annotation determines the position of the field in the filter of the List Report.
- @UI.chart: This is the main annotation regarding our demo's purpose. It basically sets the chart type, dimensions and measures for a Smart Chart consumption inside the Object Page.
- @Semantics.amount and
  @Semantics.currency: These annotations define a relation between an amount field and respective currency.
- @EndUserText.label: This annotation provides a label for a specific field.
- @OData.publish is used to publish the OData service automatically without the need to create an OData project through transaction SEGW (you can check more details about OData Project in the next section).

An interesting point is that both CDS views share the same key, so why I need to split the content in 2 different views?

I don't know exactly but there is some kind of restriction, and the Smart Template expects this specific structure of separate views (one for the main entity and another one for the chart).

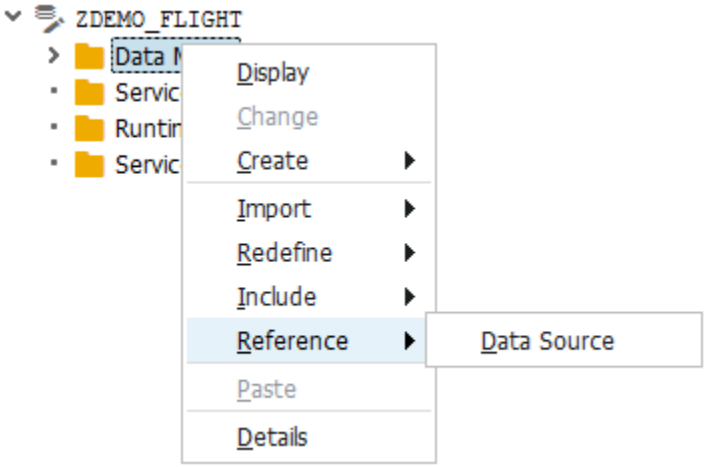
I've tried to place the chart (and respective annotations) in a single view, but the Smart Chart wasn't rendered properly by the UI5 application. You can also notice that we define the association as [0..*]instead of [0..1], if you don't follow this convention the chart will not appear in the screen as well.

Now the ABAP CDS views are finished, we just need to expose/activate our OData service and generate our UI5 application.

2. OData Project

There are 2 ways to create your OData project consuming ABAP CDS views:

Create a new project through SEGW transaction and include your CDS views by Reference. Just right click on the Data Model folder and select Reference -> Data Source.



Include the @OData.publish annotation in the header of your CDS view, the system will create your OData project automatically based on the field structure and annotations. (For now It is added in the code)

How to create ODATA Service?

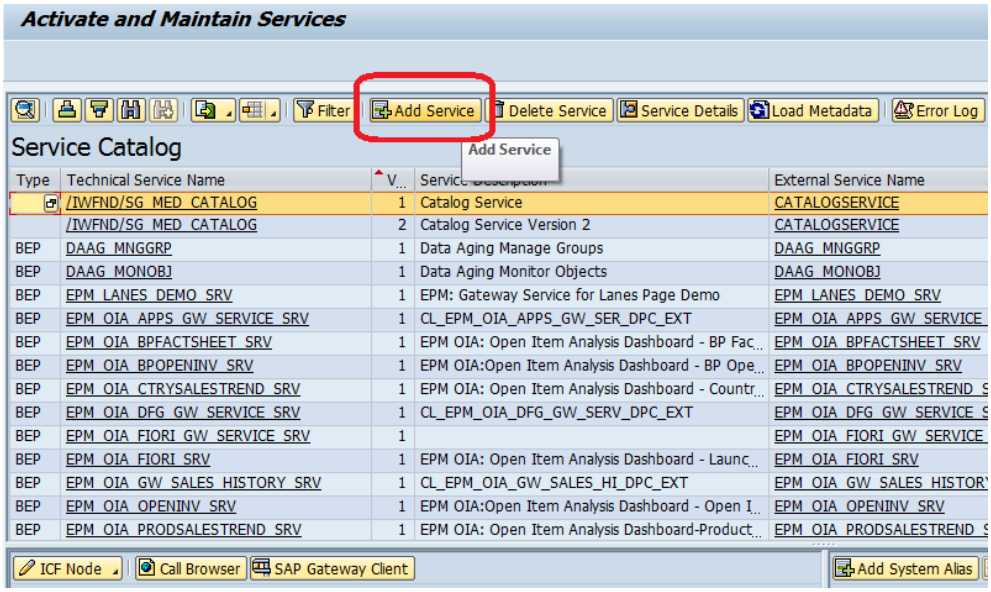
If you do not know how to create a service, please follow these steps. (Otherwise Skip this step)

1. Go to T code SEGW in SAP On-Premises.
2. Create a New ODATA service with the name Z-----.
3. Right Click on Data Model, Go to Reference, and Click on Data Source as displayed in the Last Image.
4. Select Your CDS Views and Press OK.
5. Click on Generate Run Time Artifacts Button (Red Circle Icon on Top).
6. Just Save Your Service and Go back.

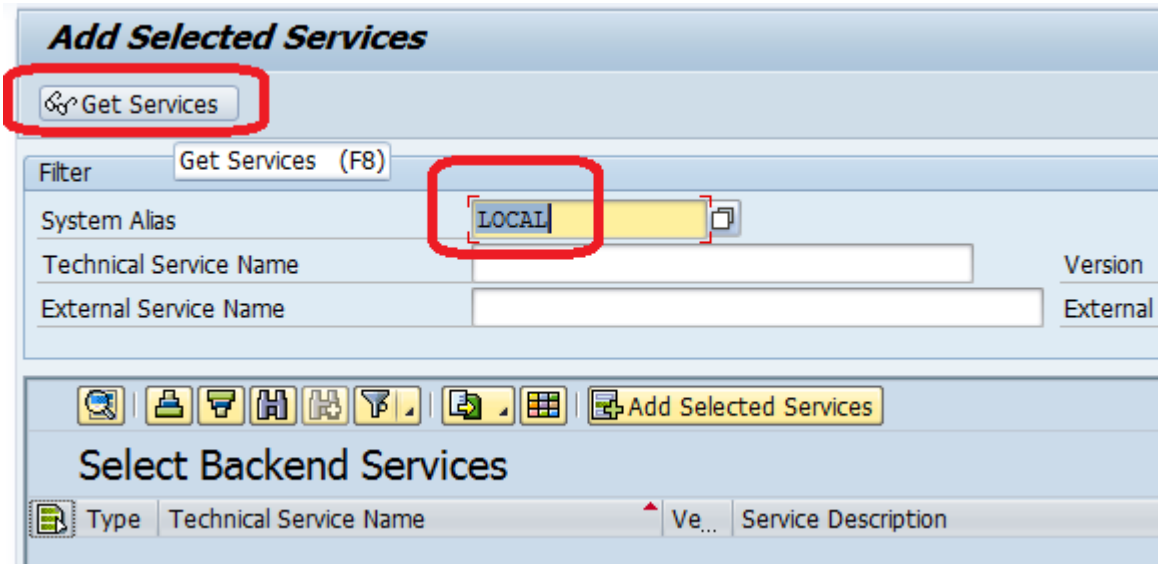
How to Register an ODATA Service?

After Creating ODATA we need to register Our Service.

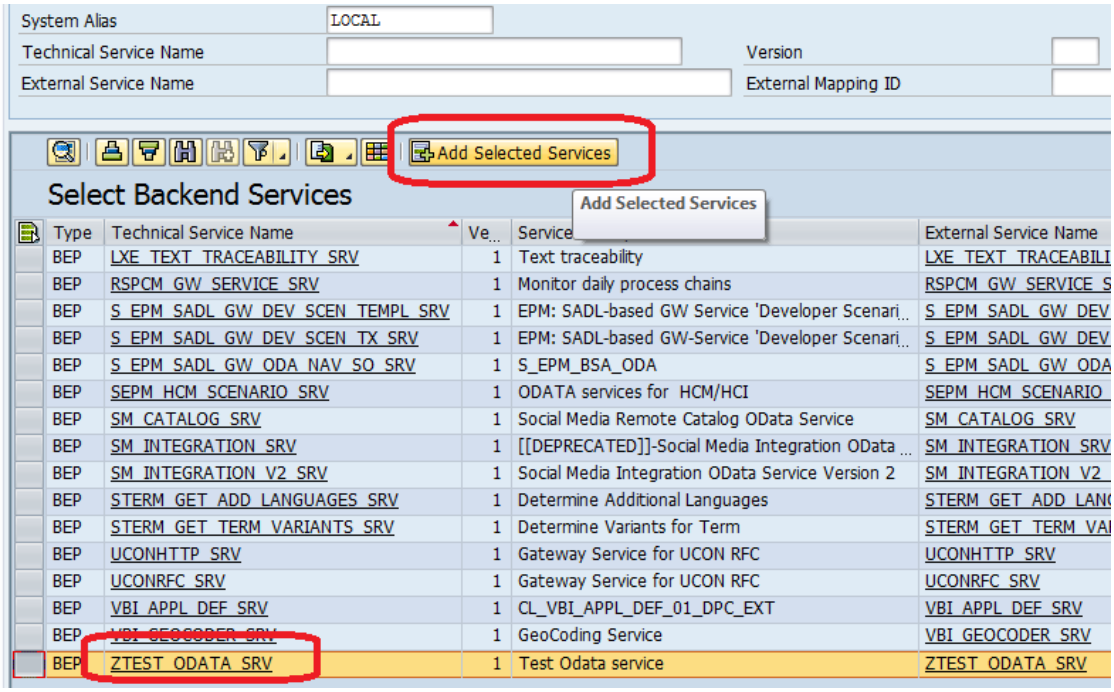
1. Enter T Code **/IWFND/MAINT_SERVICE**
2. Opened windows show the list of registered services.
3. Click on the button 'Add Service' to proceed with new service registrations.



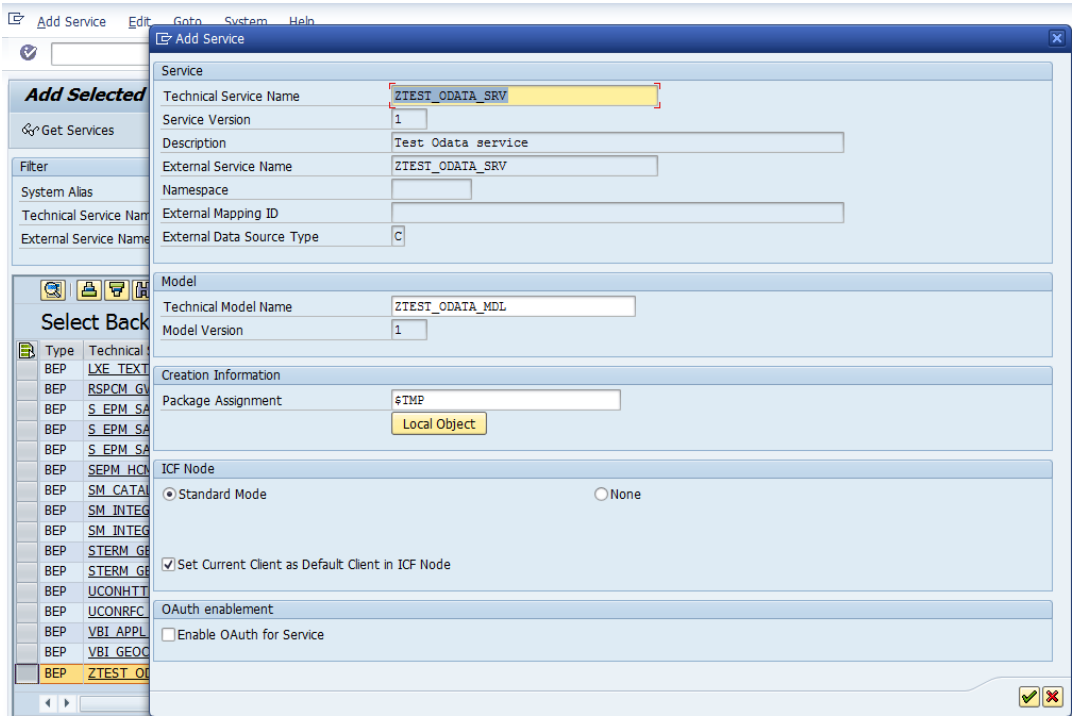
4. In the next window, get an unregistered service list.
5. select System Alias 'Local' -> click on the button 'Get Services'
6. You can also write ODATA name in Technical Service Name by putting * at the end of the name, for example "Z-----*" to get the specific one.



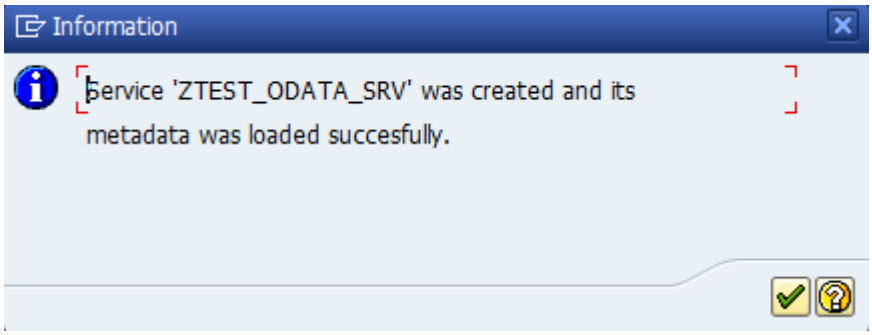
7. From the unregistered service list, select the desired service.
8. and click on the button ‘Add Selected Service’.



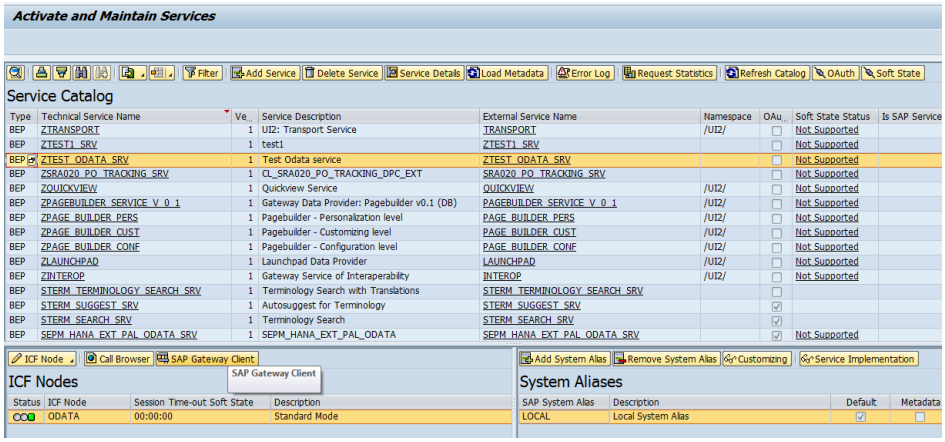
9. In appear pop-up click select input details as shown in below scree and here enters your desired package, for example purpose $TMP



10. Click ok in next pop-up.



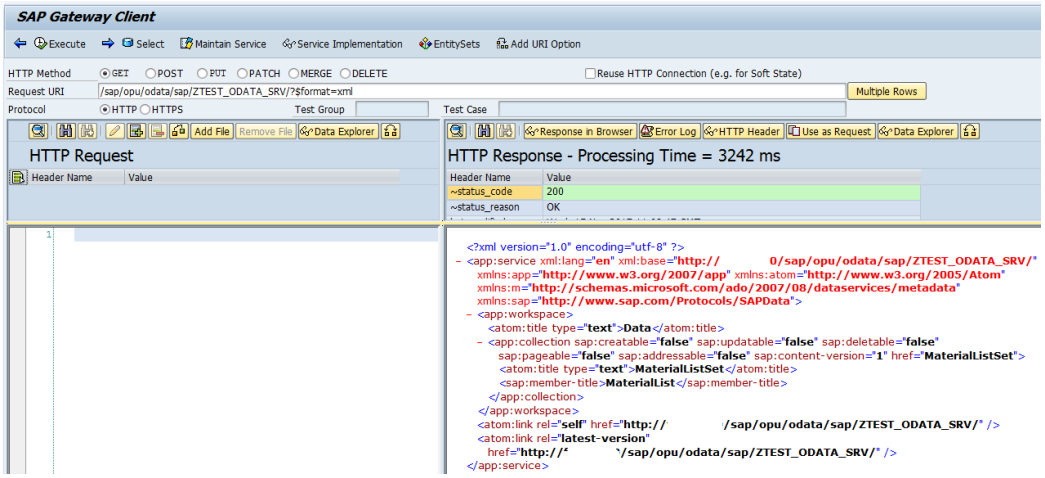
11. Once the service gets registered, we can view it in t-code '/n/iwfnd/maint_service' within the registered list.

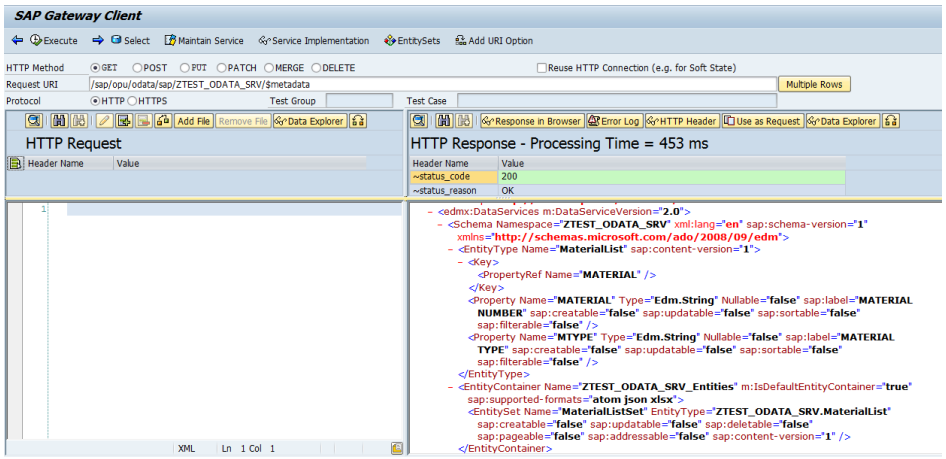


12. For testing the service, either select the service as in the above screen and click on the button 'SAP Gateway-Client' or go to t-code '/n/iwfnd/gw_client'.

Test OData Service

1. Just Click on SAP Gateway Client and Select Entity Set from EntitySets Button.
2. Selected Method Get and Press Execute





3. Status Code ‘200’ and reason ‘OK’ with green color response layout represents everything is ok.

NOTE:

We use the second option to publish our service, but no matter the approach you decide to follow just remember always to activate the OData service in the Front-end server (SAP Gateway server) through the transaction /IWFND/MAINT_SERVICE.

Create Fiori APP

Now, The ODATA Service is Created and Published.

Let's Create Fiori App now.

1. UI5 Project (Web IDE)

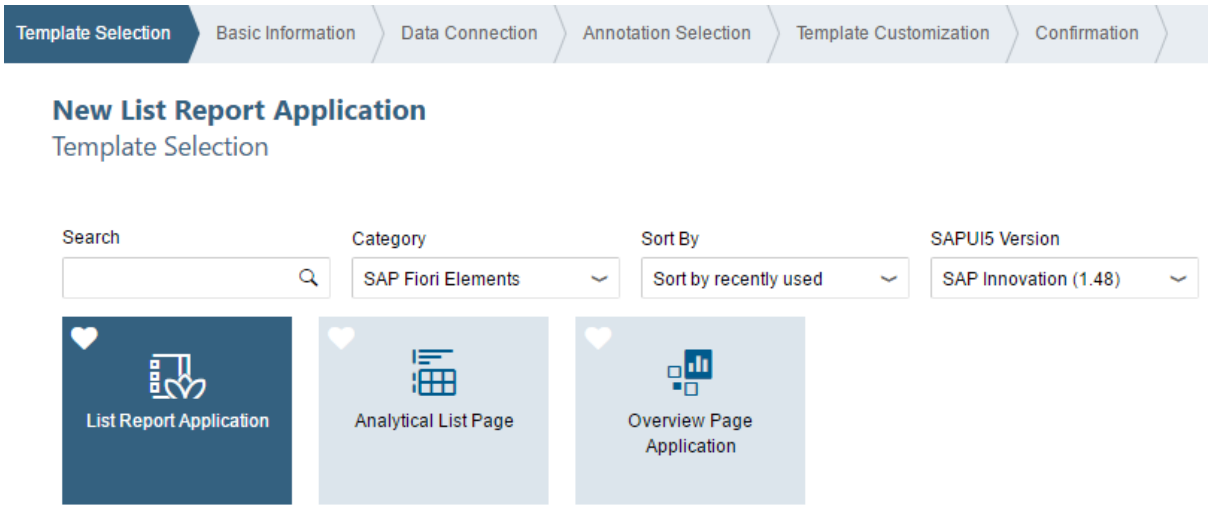
There are some types of annotations that are not available through the ABAP CDS, in this case we need to mix a little bit of the local annotations (published inside the UI5 application) with the annotations generated by the ABAP CDS views.

I, personally, prefer to include all annotations in the CDS views because if I need to execute a maintenance there is no need to re-deploy the whole application, we just need to transport the ABAP CDS view that holds the relevant annotations, and the job is done!

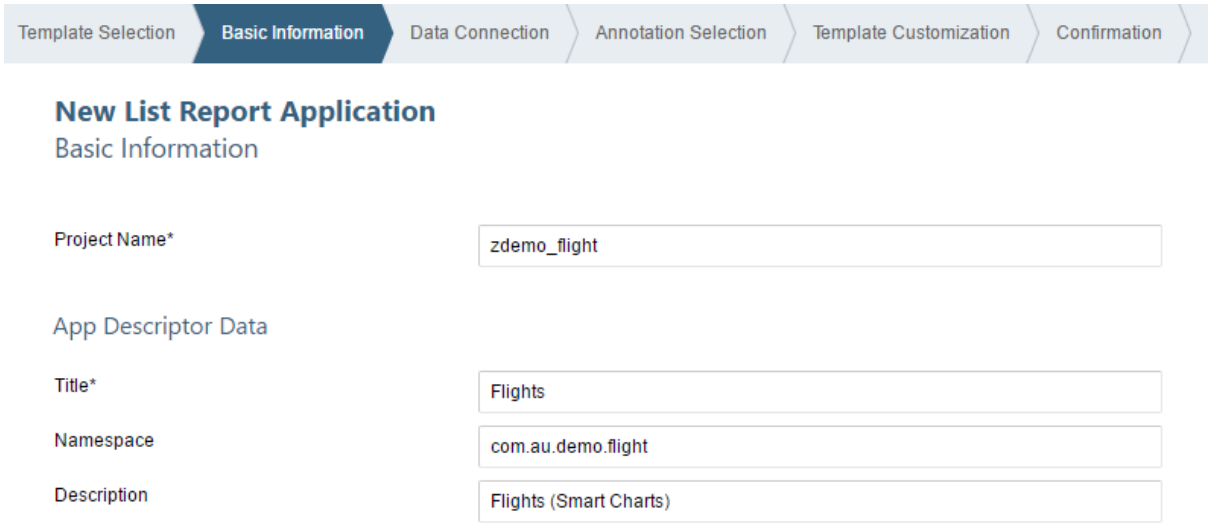
But in the case of a Facet configuration (Object Page sections), there is no other option instead to configure through the UI5 local annotation.

Let's start creating a new project based on a List Report Application.

Note #1: I'm using the SAP Innovation version 1.48, but Smart Charts and List Report Application are available since the version 1.44, if you are working with an on-premises solution you can still use this functionality.

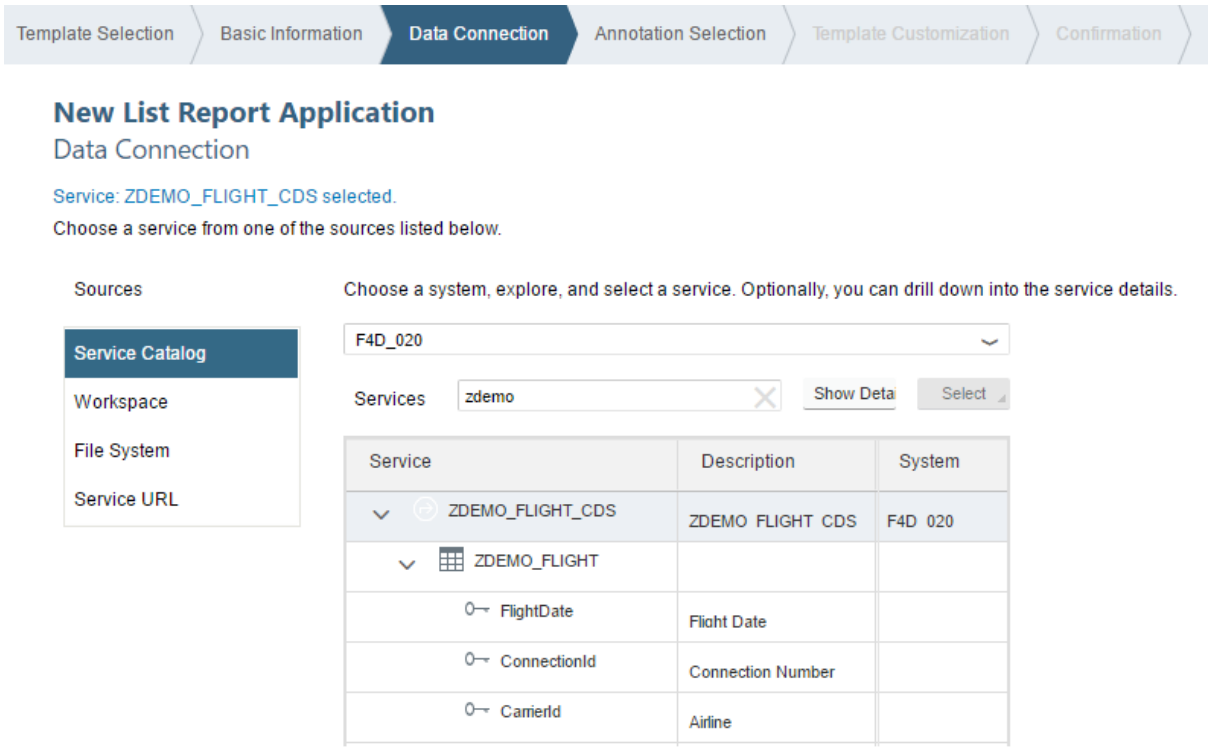


Fill the project name, title, namespace, and description:



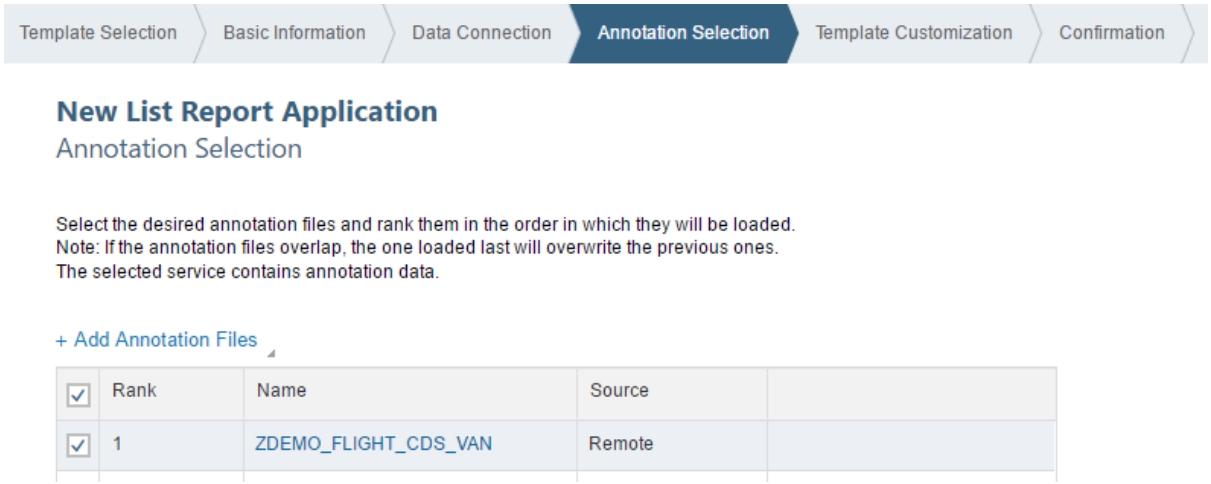
Define your data source and select the ZDEMO_FLIGHT_CDS service.

Note #2: Since we are publishing our OData service through the @OData.publish annotation, the system generates a project with the name of our ABAP CDS view + the _CDS suffix.

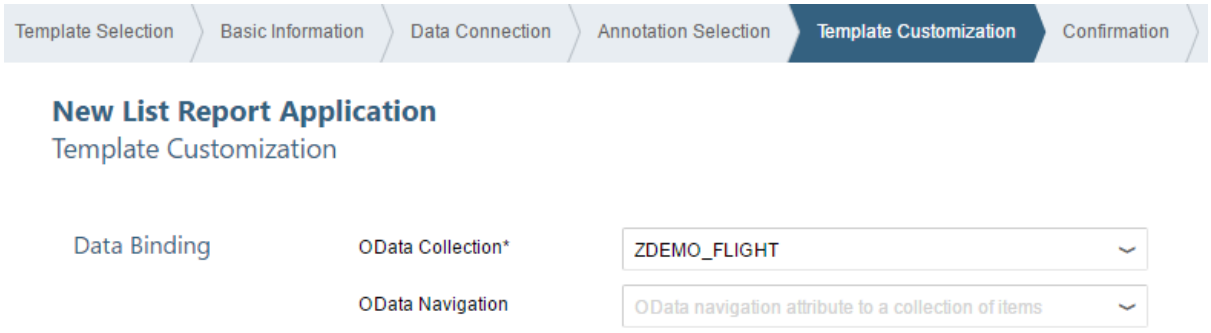


Select the annotation ZDEMO_FLIGHT_CDS_VAN (generated automatically by the OData Service / ABAP CDS).

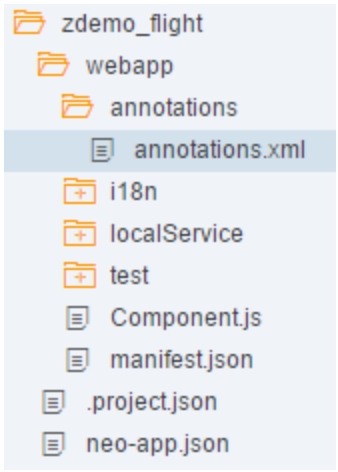
Note #3: If you don't select this option all the annotations declared through your ABAP CDS view are not going to flow to the UI5 application.



Finally select your OData Collection ZDEMO_FLIGHT and confirm the template creation.

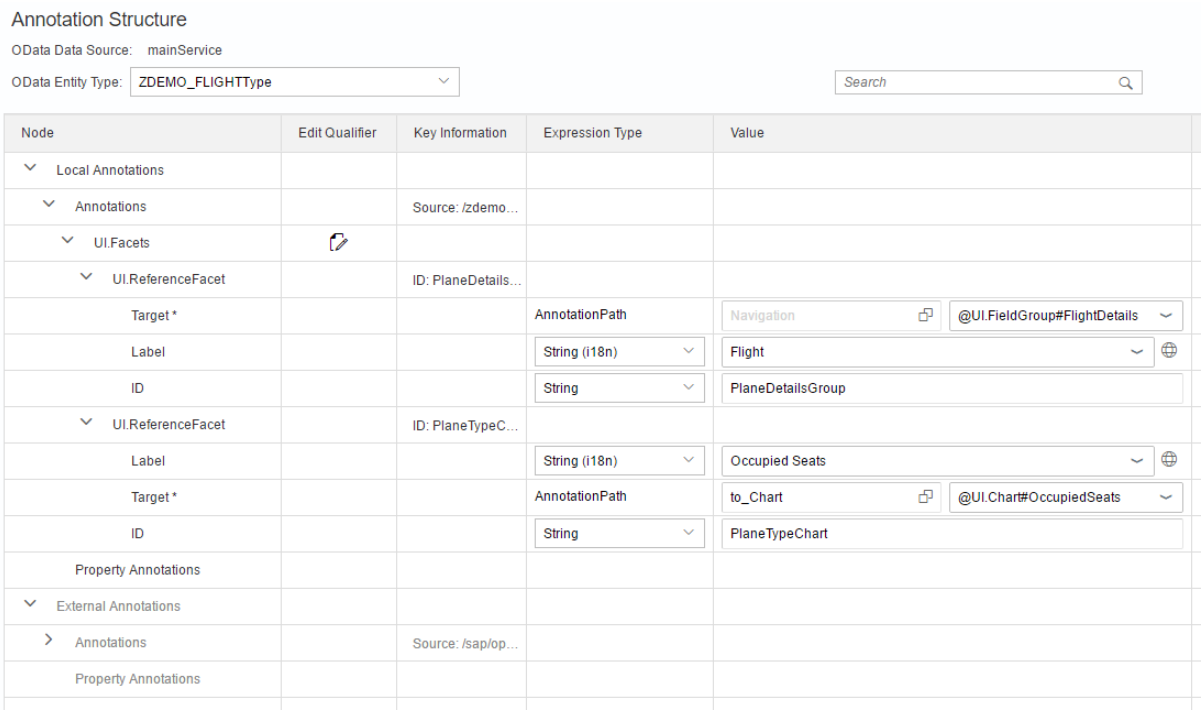


Now open the project path, annotations folder and the file annotations.xml.



Open the annotation modeler and select the Entity Type ZDEMO_FLIGHTType.

Let's create 2 new facets, for the first one we will reference the field group with the ID FlightDetails , and in the second facet we will point to the chart annotation with the ID OccupiedSeats.

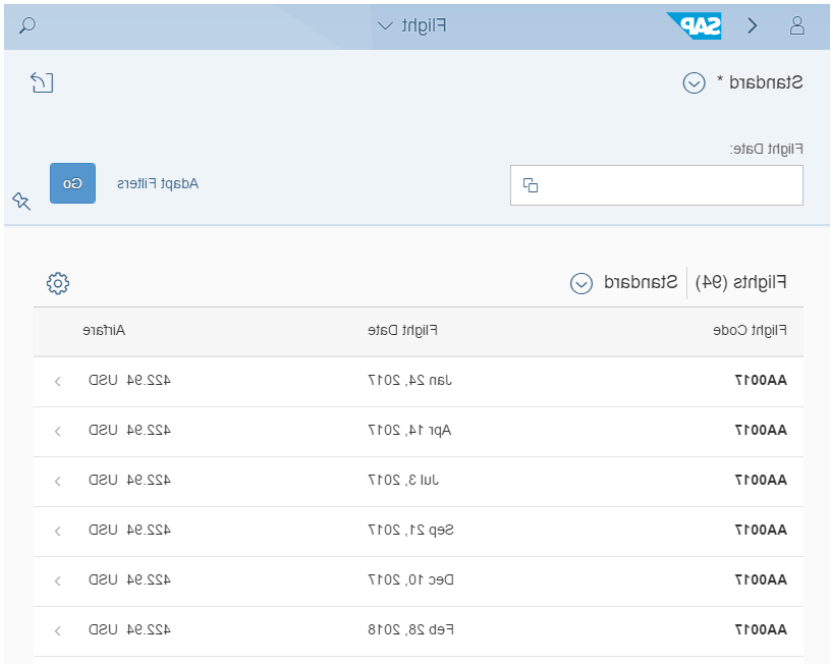


Note #4: Notice that all annotations that we declared through our ABAP CDS views are available under External Annotations section, the Annotation Modeler automatically merges annotations from external sources (ABAP CDS or OData project) with the local annotations declared inside the UI5 application. If you want to declare your Smart Chart in a scenario without ABAP CDS there is no restriction, you just need to open the Annotation Modeler and place your own @UI.chart annotation.

After you finish to edit this file, save the content and start the application.

This is the expected outcome:

List Report:

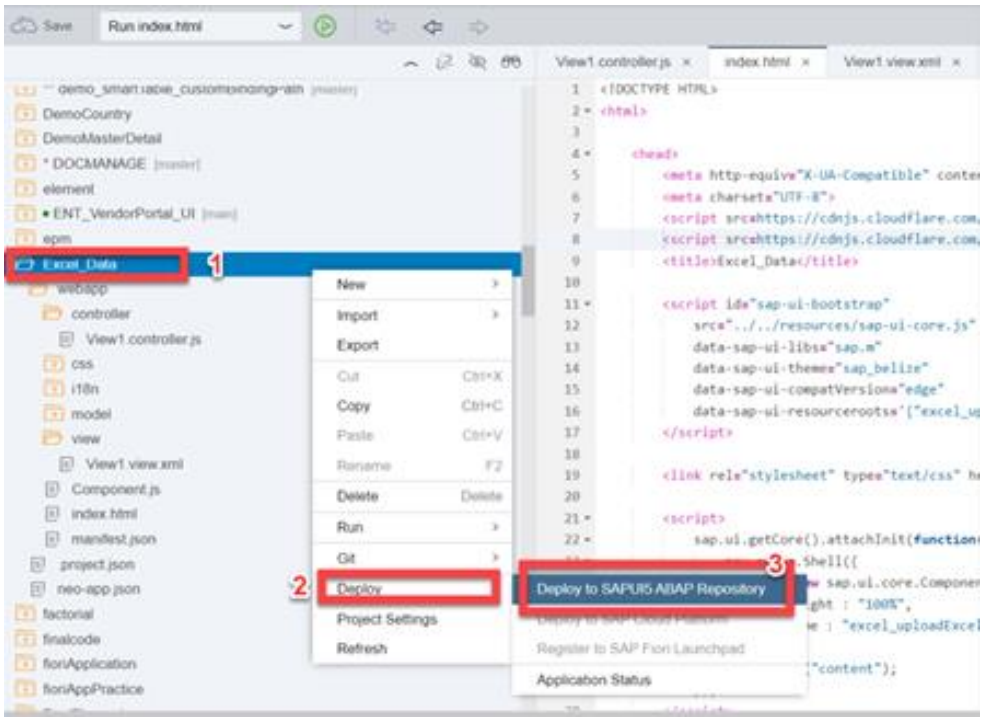


Object Page:


Flight
AA0017 747-400
Flight
Occupied Seats
Flight Code:
AA0017
Flight Date:
Apr 14, 2017
Occupied Seats
Details
Drill Down
Occupied 1st & Occupied bus. & Occupied econ. &...
400
300
200
100
0
Apr 14
2017
AA0017 / Occupied 1st
AA0017 / Occupied bus.
AA0017 / Occupied econ.
AA0017 / Max. capacity e...

Deploy the Fiori app using WEB-IDE:

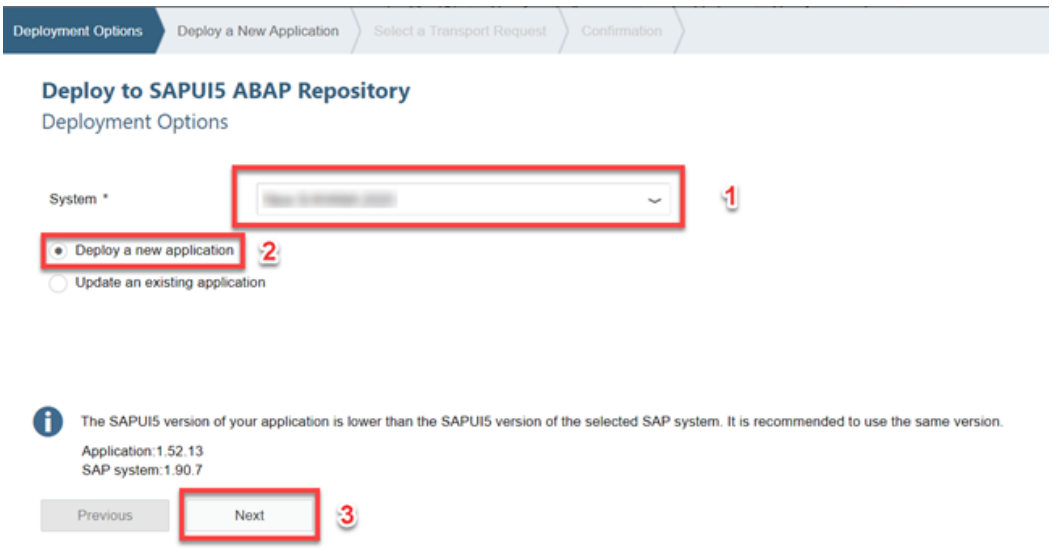
1. Go to the WEB-IDE, right-click on your project –> Select Deploy –> Choose your System (Deploy to SAPUi5 ABAP Repository)



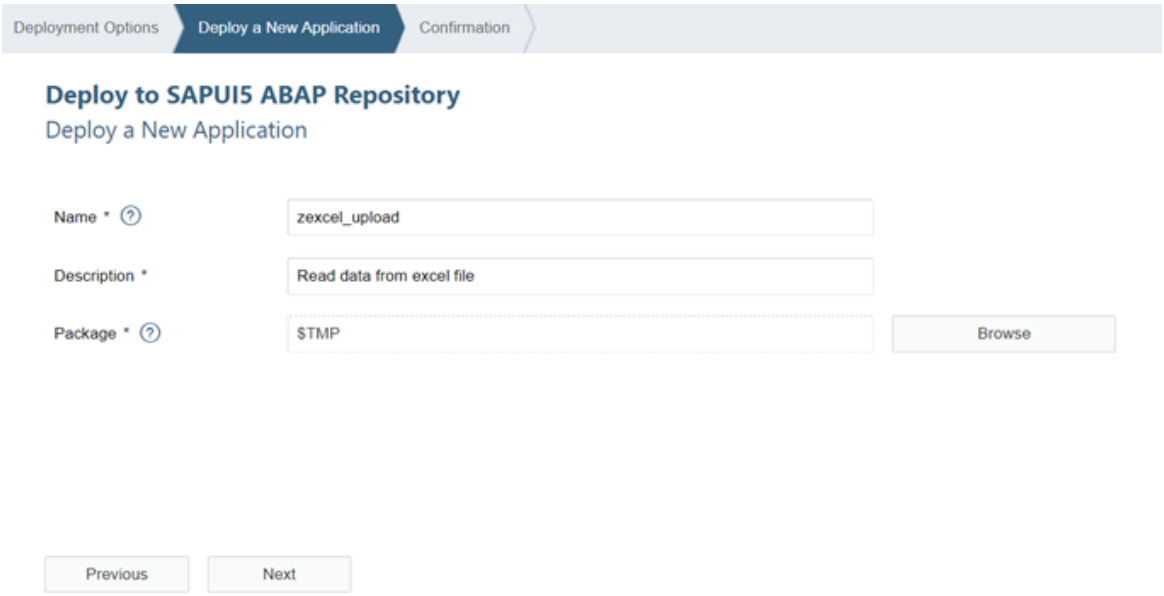
2. You will be on the next page (Deployment Option), Here select your required ABAP or Cloud system –> Next.

Note:

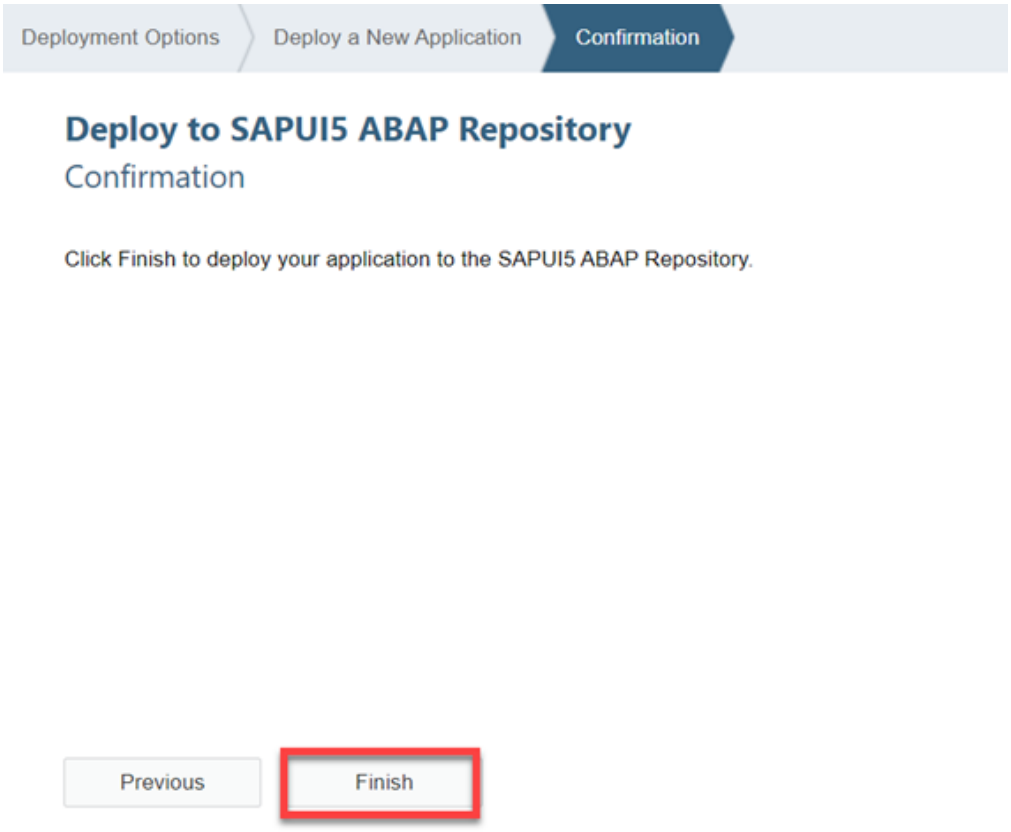
If you are deploying for the first time, select Deploy a new application. Else application is already deployed, select update an existing application.



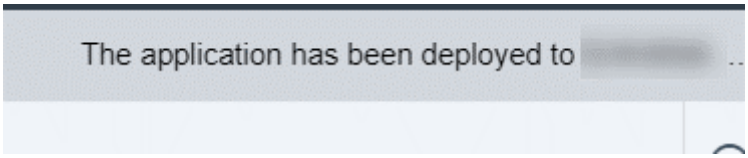
3. Provide the application Name, Description, and Package, in – Deploy a New Application Screen and select Next.



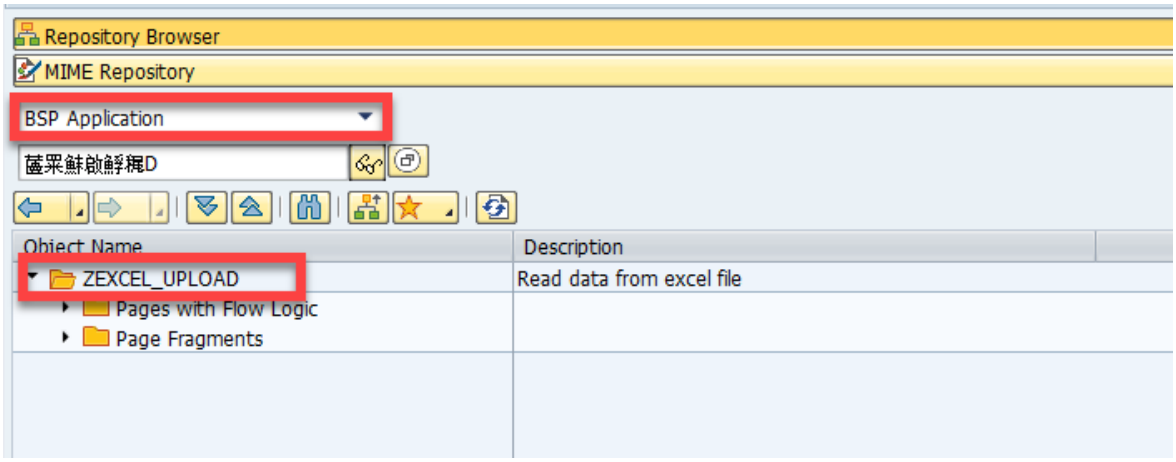
4. If the selected package is local, Choose Finish. If it requires TR, select TR for your application.



A notification message displays once the application is deployed successfully.



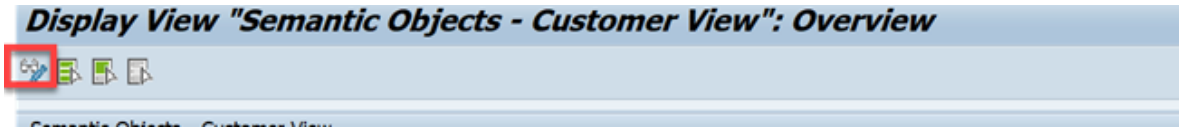
5. Go to SAP Logon Pad –> Tcode SE80 –> Select BSP Application from Dropdown –> find your deployed application.



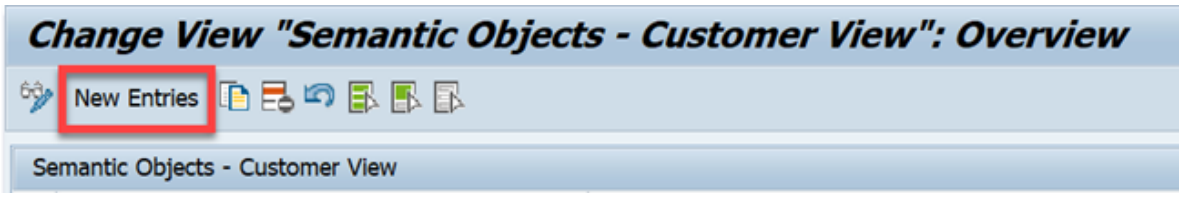
Steps to create a Tile and Target mapping in Fiori launchpad Designer

1. Create Semantic Object using T-Code – /N/UI2/SEMOBJ.

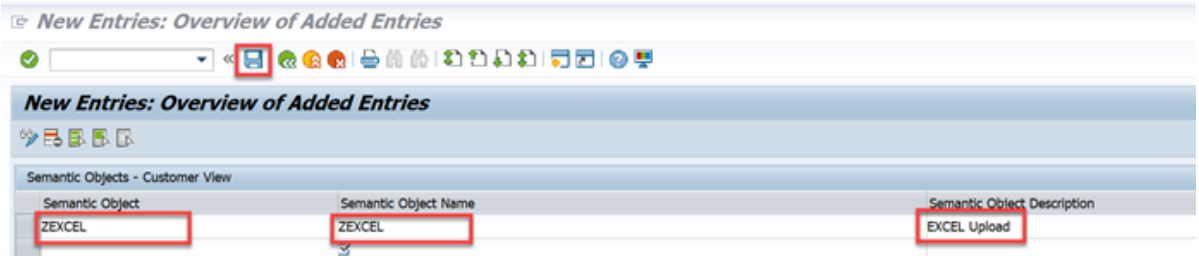
   Click on Edit Button



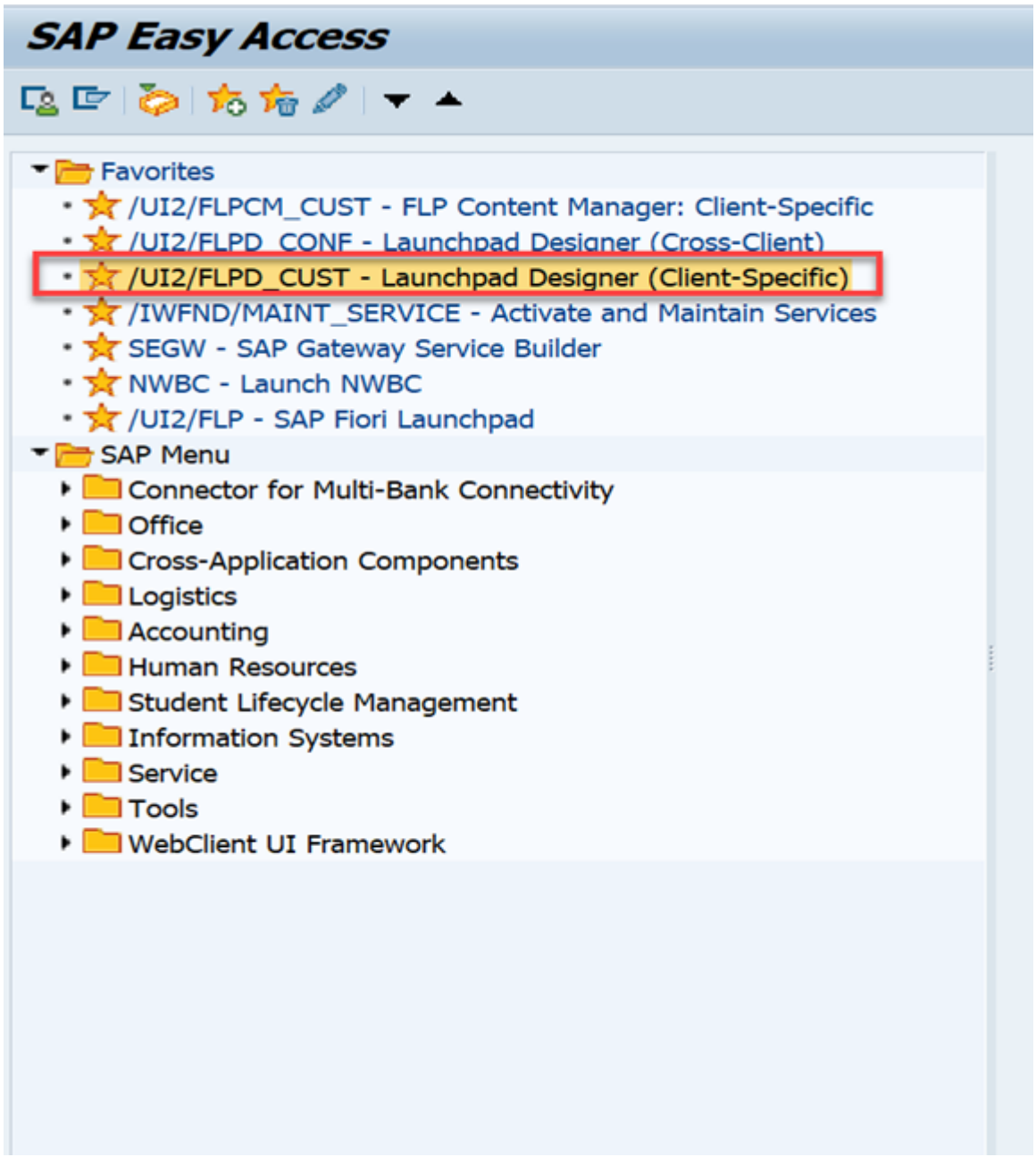
2. Select New Entriere.



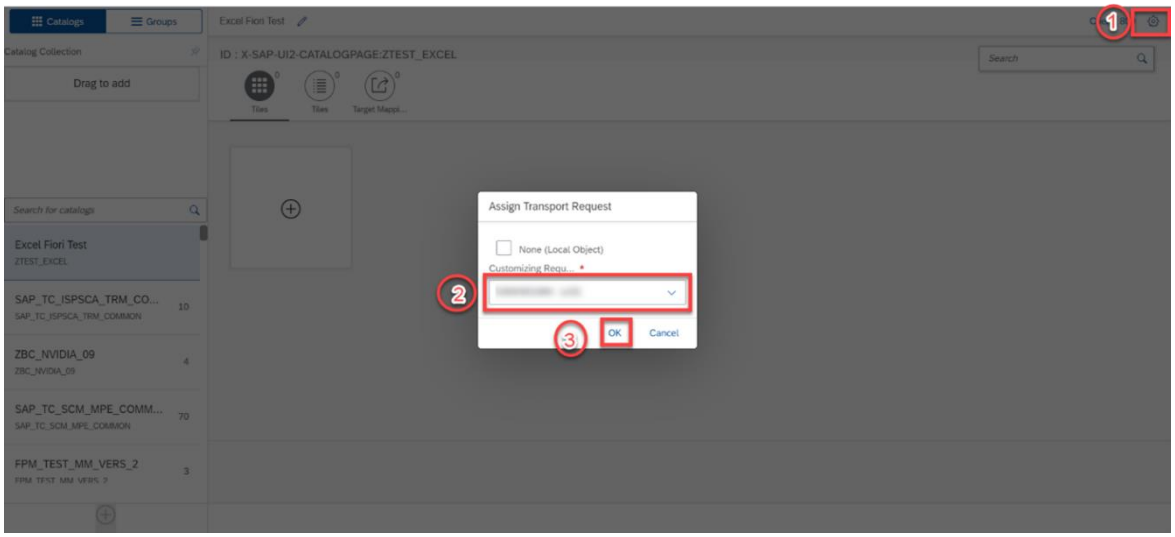
3. Provide the new Semantic Object, Object Name and Description and Click on Save Button.



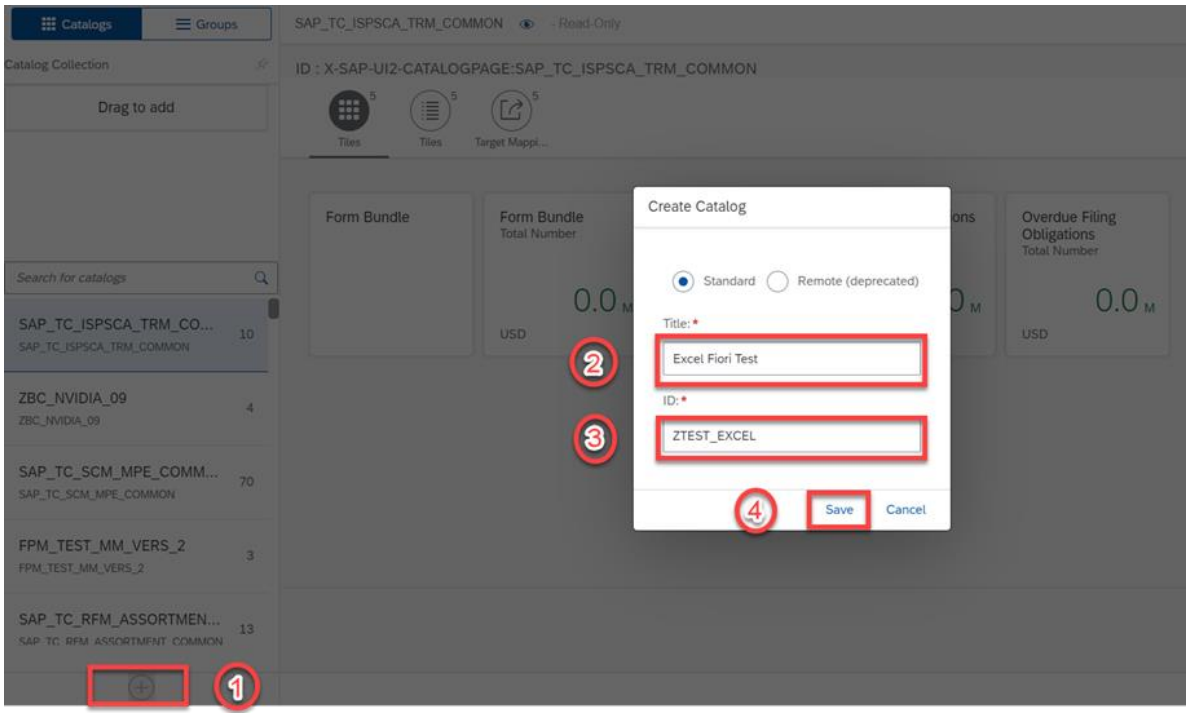
Go to SAP LogOn –> Execute the TCode – **/UI2/FLPD_CUST** (Launchpad Designer).



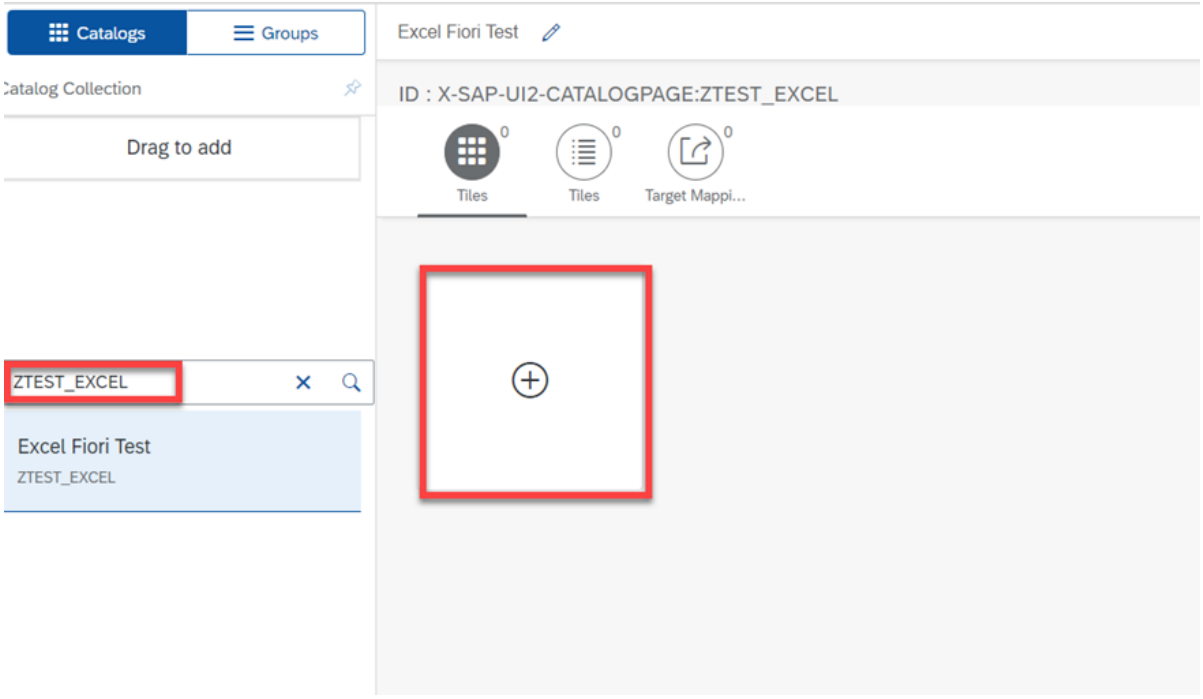
Click on Setting Button –> Select the TR –> choose Ok.



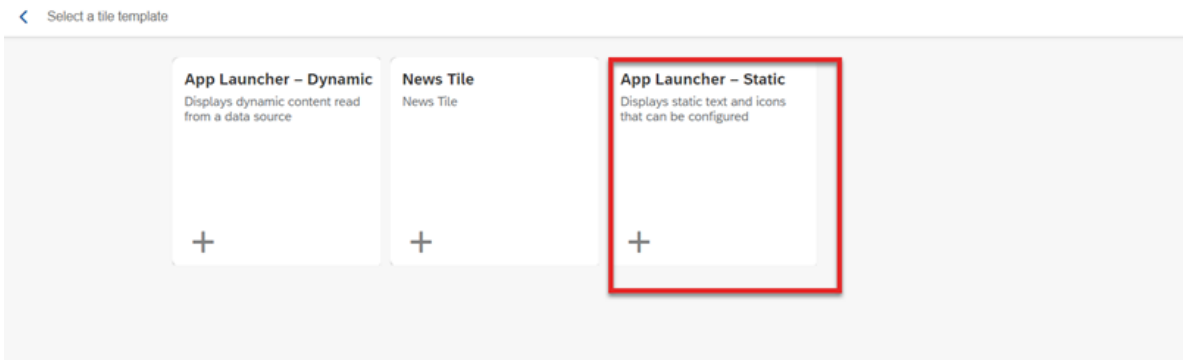
If catalog information is not available, create a catalog –> Click on Plus Button –> provide the required information –> click on Save.



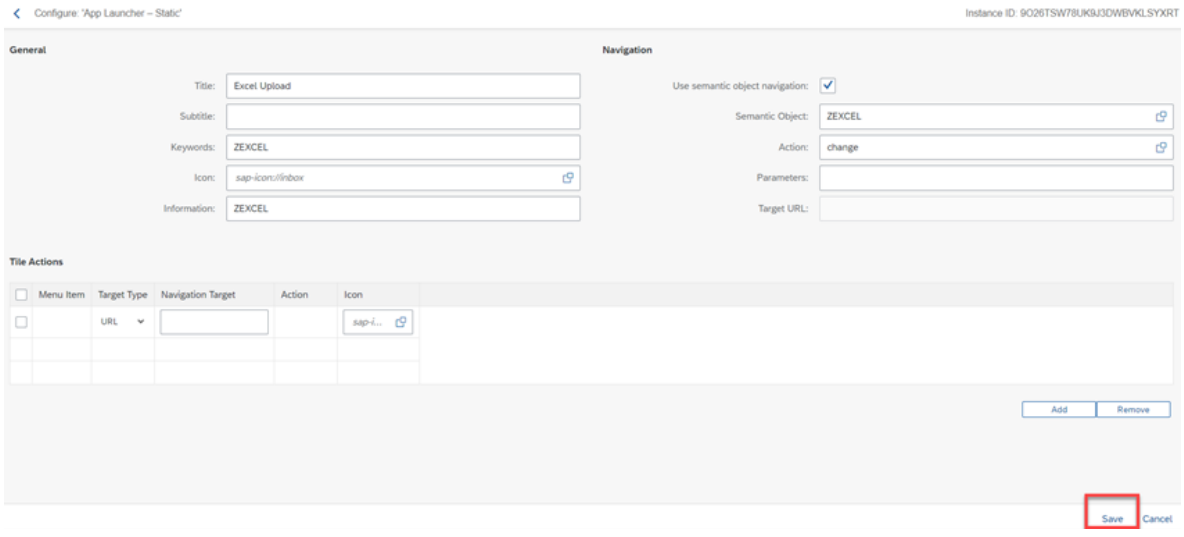
If catalog already exists, select the catalog –> Click on Add tile plus Button.



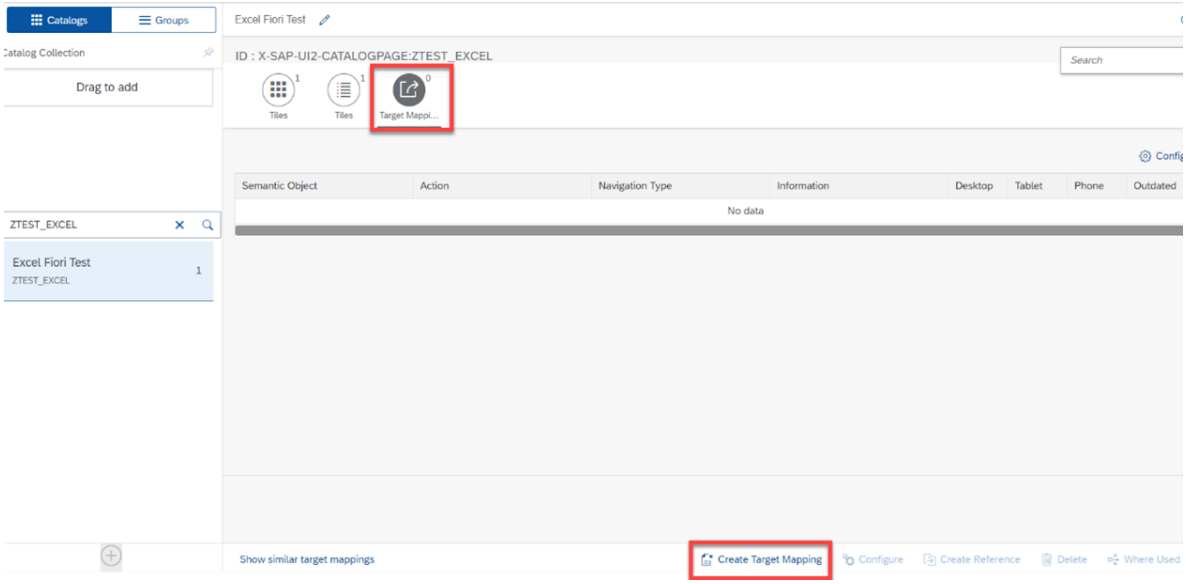
Select a Tile template, here I am going to select App launcher – Static.



Provide the Title, information, Semantic Object, Action and other required Information Click on Save Button.



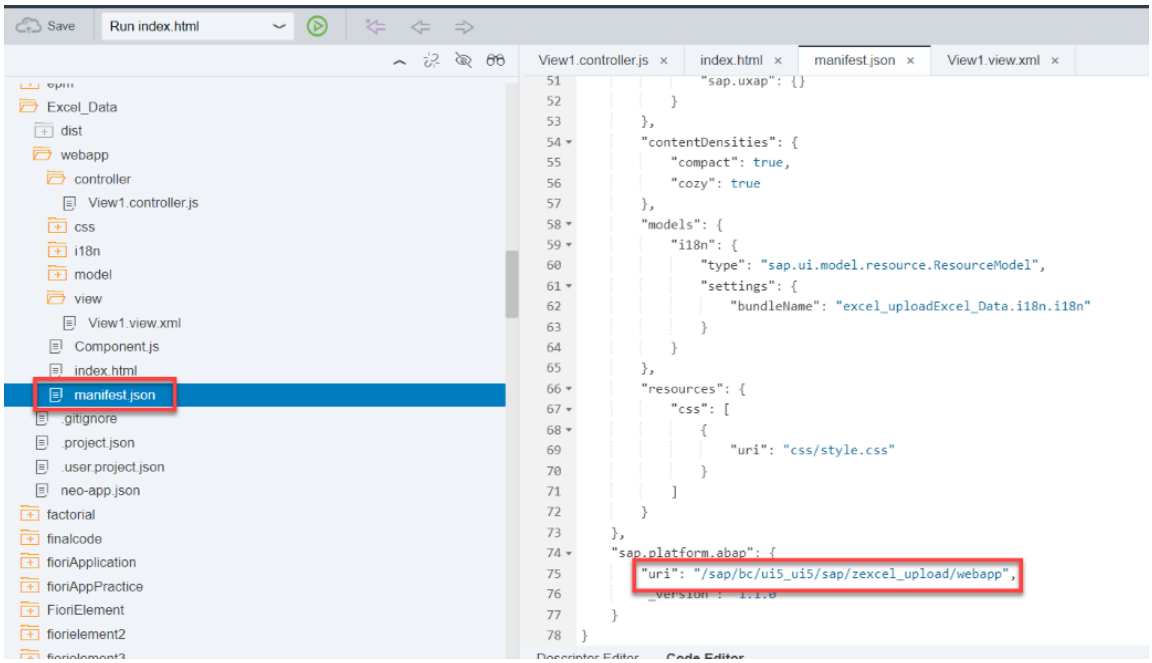
Create a Target Mapping à Go to Target Mapping à Click on Create Target Mapping.



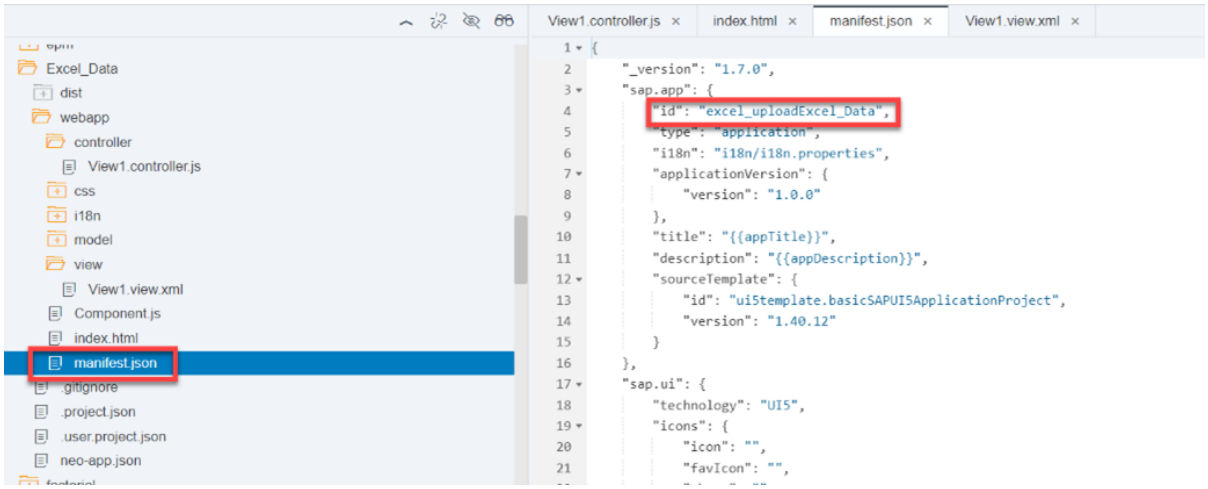
Provide the required information Semantic Object, Action, Title, URL, ID and Click on Save Button.

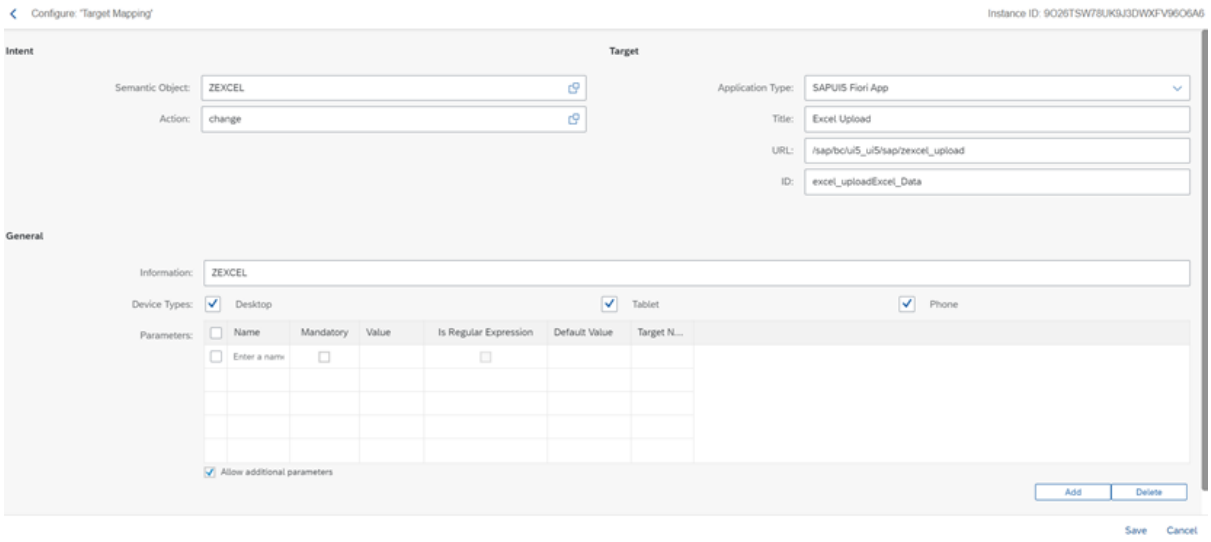
Note:

How to get URL –> Go to the WEBIDE –> your project –> manifest.json file –> Find the URL info.



How to get ID: Go to the WEBIDE –> your project –> manifest.json file –> find the ID.

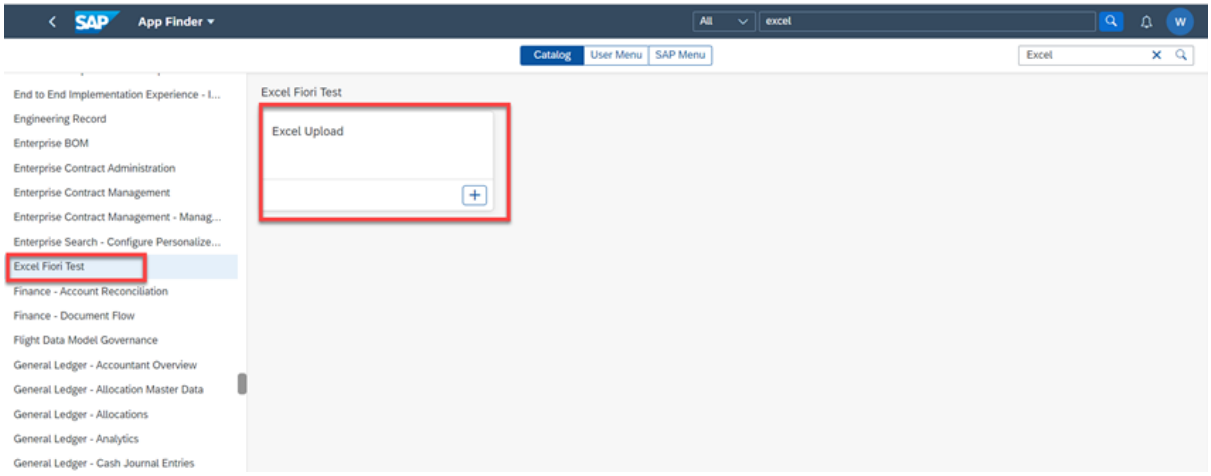




The Fiori app tile has been created successfully. Go to Fiori launchpad –> App Finder –> Search your application.

Note:

If the Fiori app is not found on Fiori Launchpad à we must assign the PFCG Role for catalog and group and assign that role to User.



Conclusion:

The Fiori app has been deployed on SAP ABAP repository. The Fiori app tile has been created.