



SAP
Development



SAP CDS to Fiori

Create CDS Views and Use in Fiori App

Content:

- 1) Install Java Version**
- 2) Install Eclipse**
- 3) Add ABAP environment for CDS Development**
- 4) Create CDS View**
- 5) Create ODATA Service**
- 6) Register ODATA Service**
- 7) Create Fiori App through Annotations**
- 8) Deploy Fiori App Through WEB-IDE**

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:

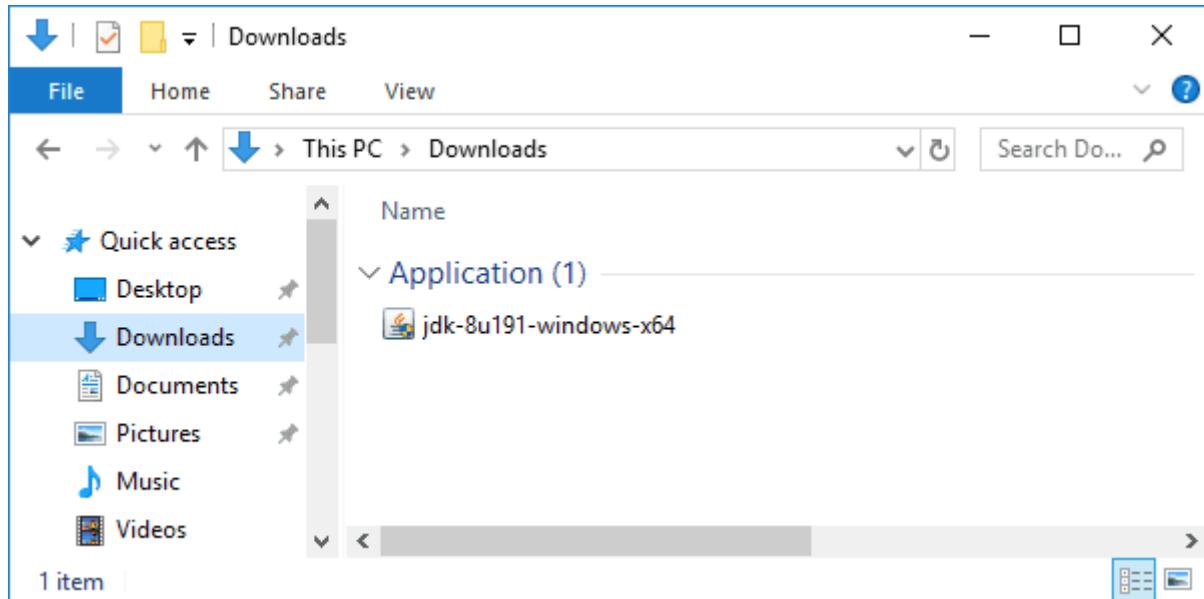
ADT is validated and tested against Java versions.

NOTE:

Please Skip This Installation If you already have Java Installed.

Install Java Version

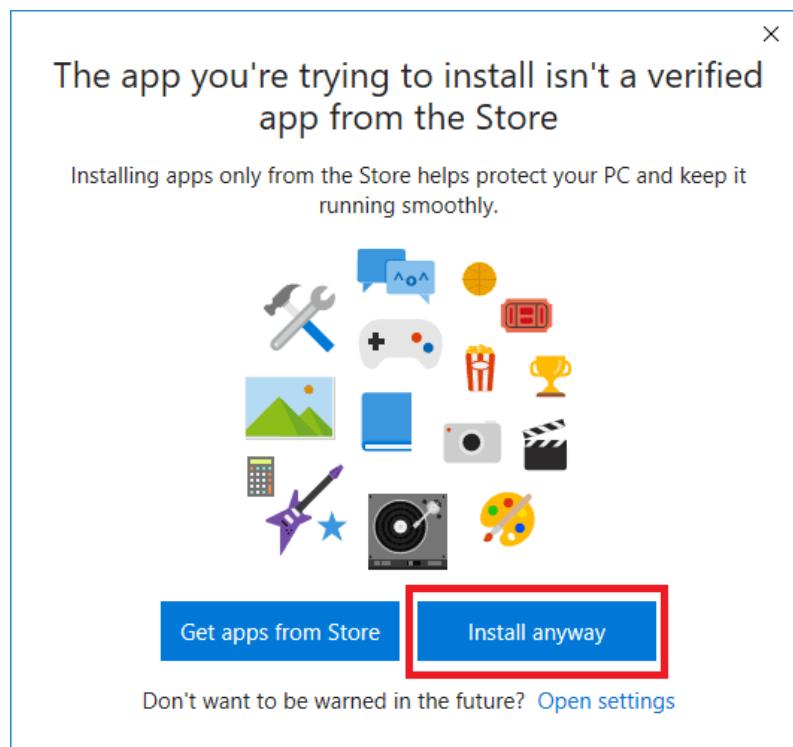
Open the location of the downloaded executable.



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.

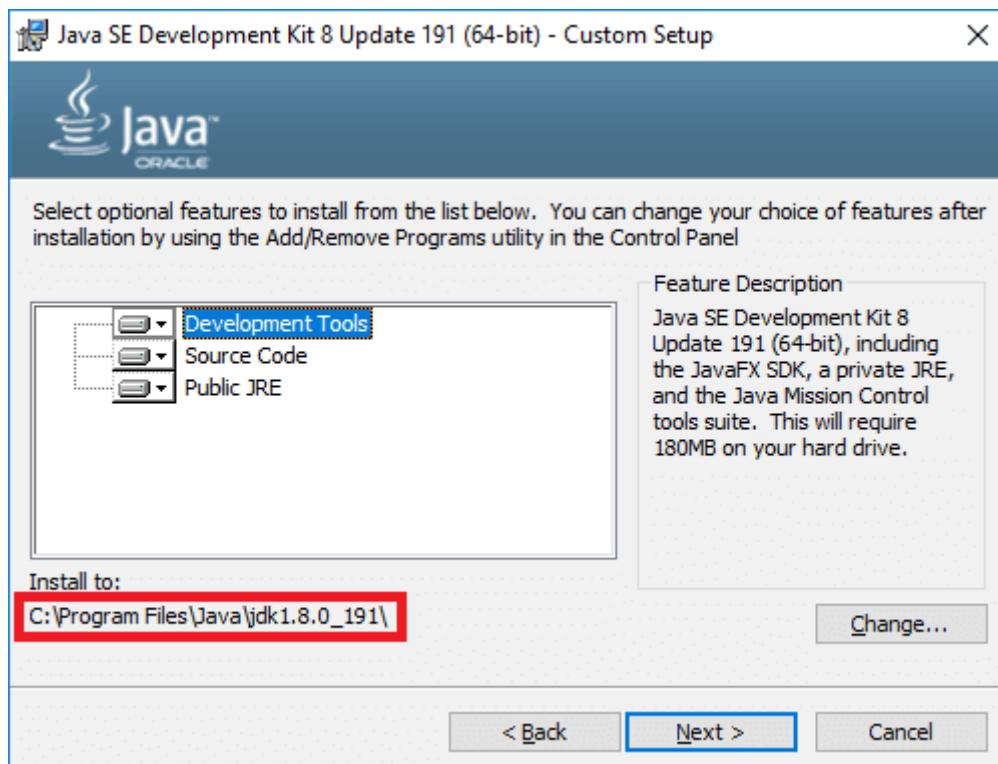


The JDK installer will start. Click **Next**.



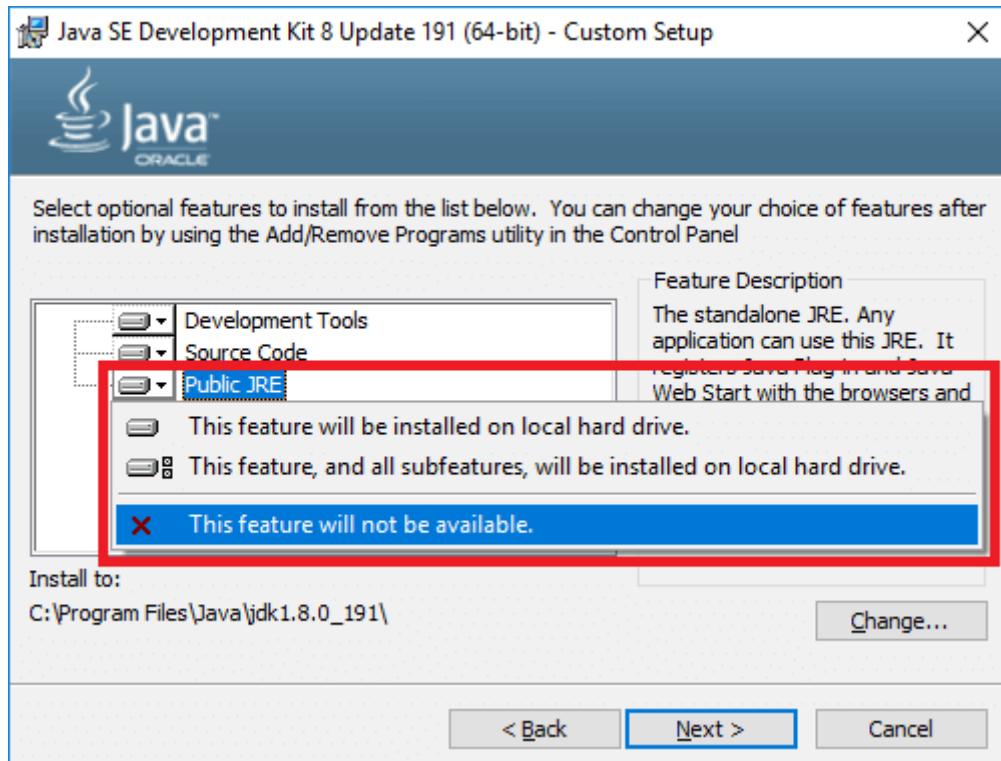
You can change the installation location by clicking on the **Change...** button.

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]**.

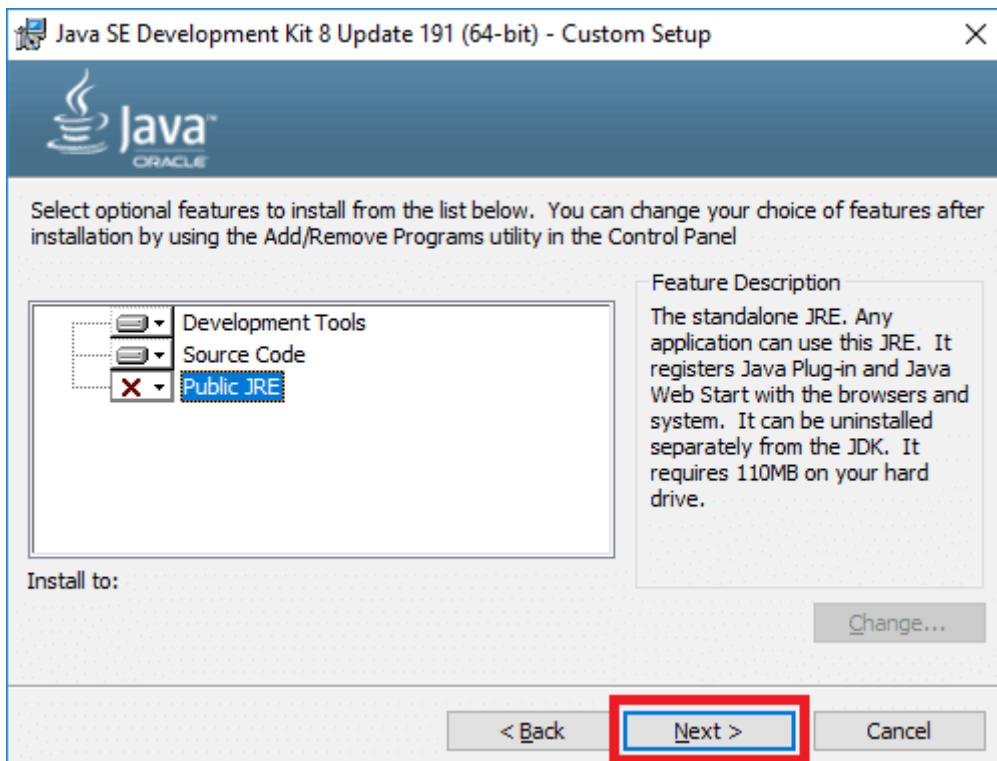


We will not install the public JRE as the JDK development tools already include a private JRE.

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

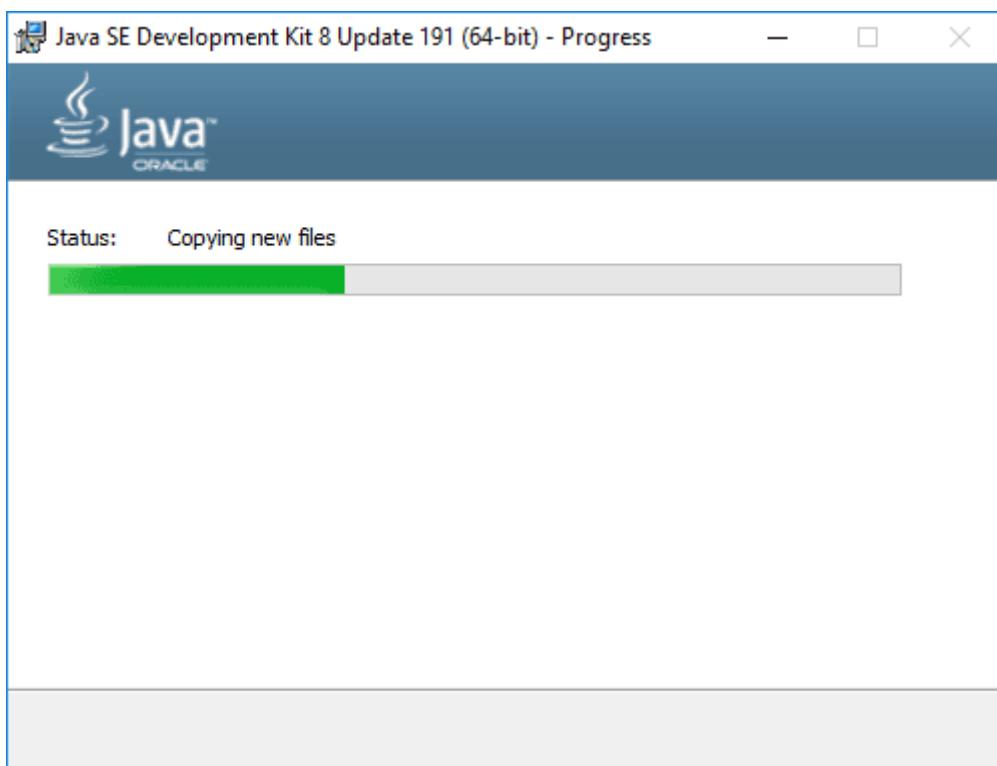


Click **Next** to start the installation.

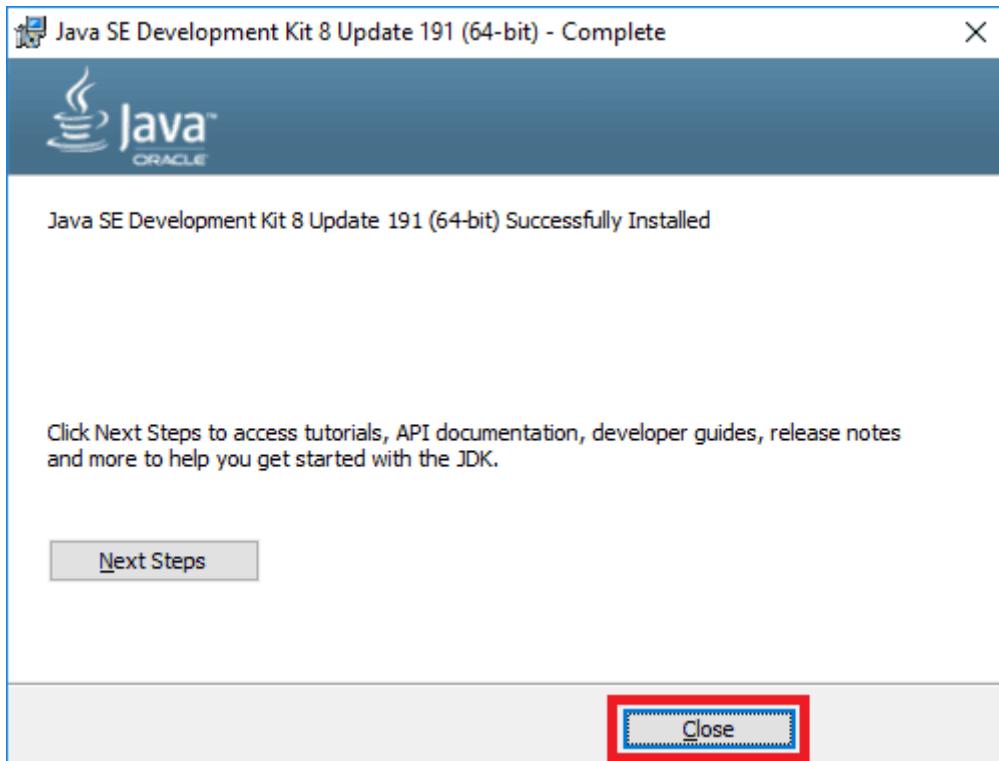


The JDK installation will now start.

A progress bar shows the various steps that are executed.



Once the installation is complete, click **Close**.

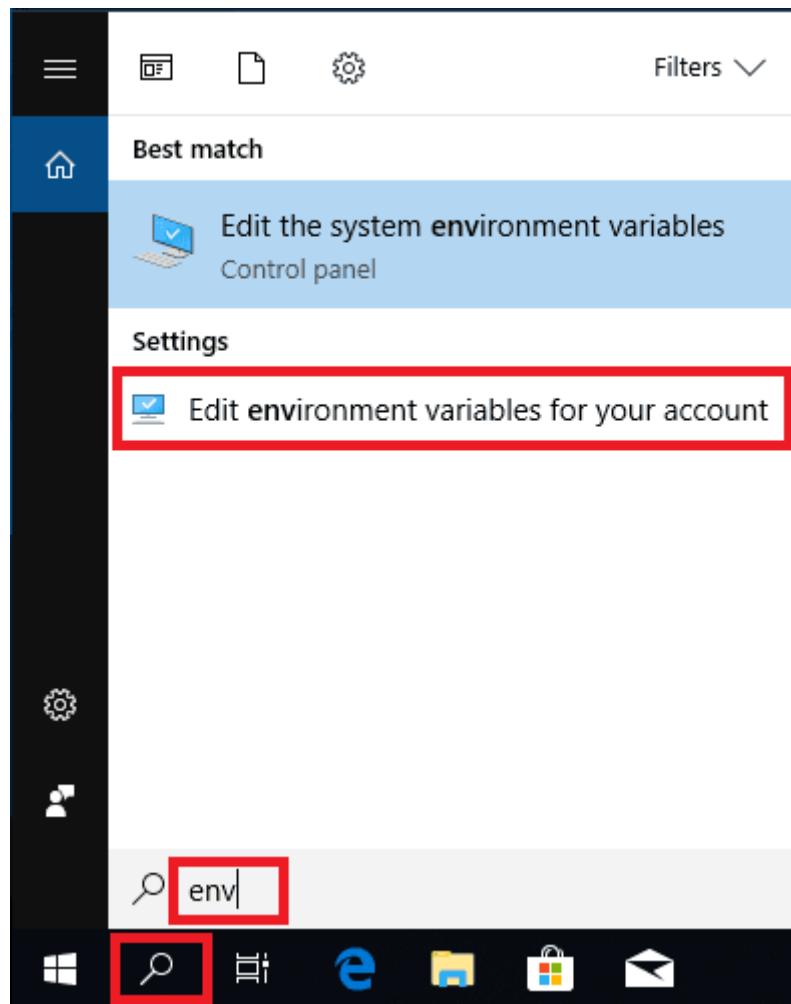


We need to set up an environment variable that will point to our JDK installation.

Click on the search button. Then type “**env**” (without quotes).

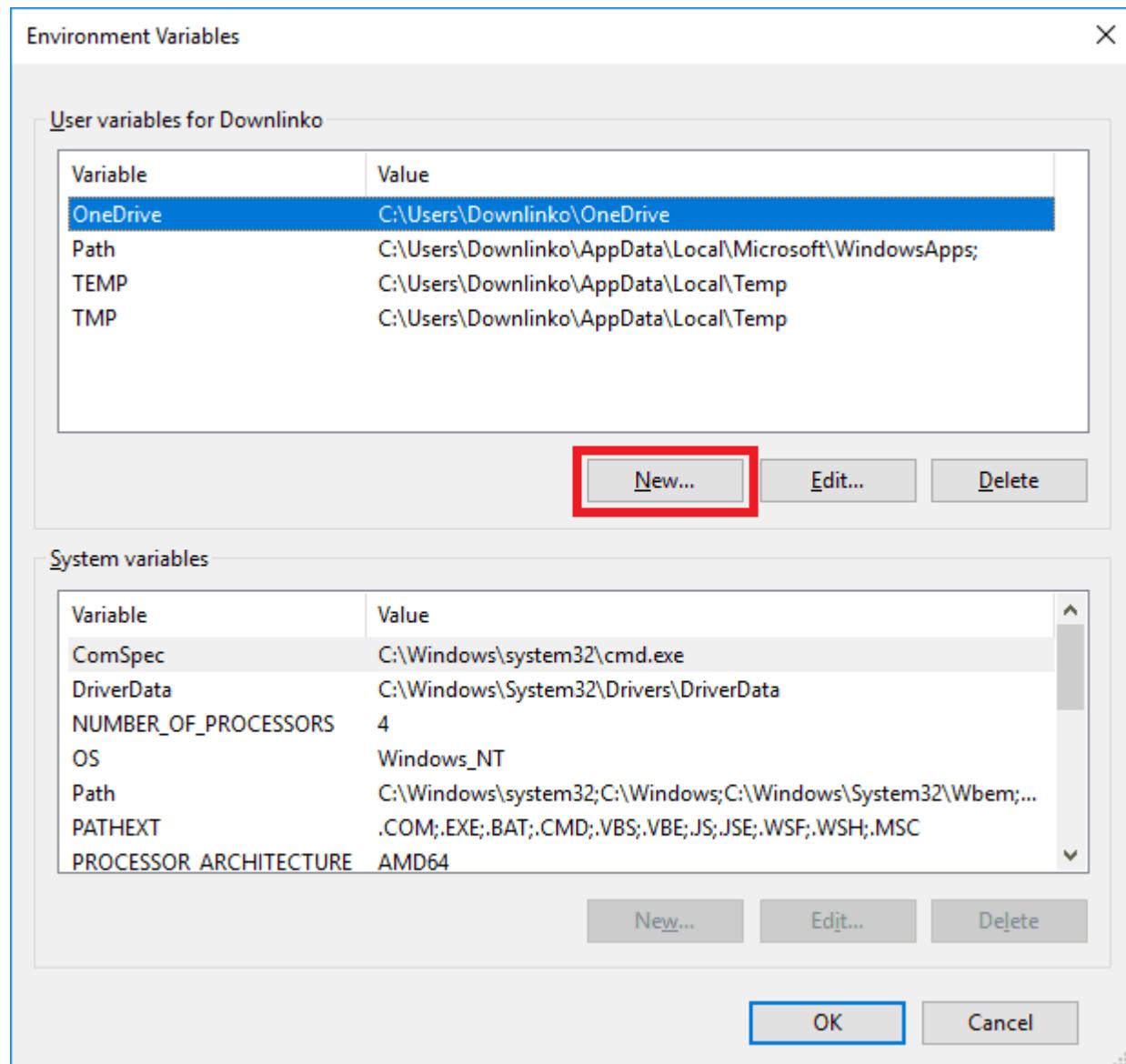
On Windows 7 click on the Windows button.

Click on the **Edit environment variables for your account** shortcut.



Wait for the environment variables window to open.

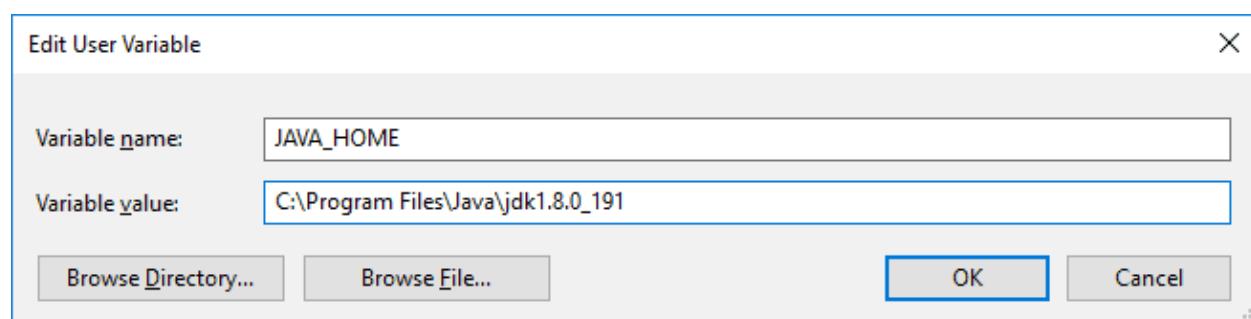
Click on **New...**.



Enter “JAVA_HOME” as variable name. Enter the [JAVA_INSTALL_DIR] as variable value.

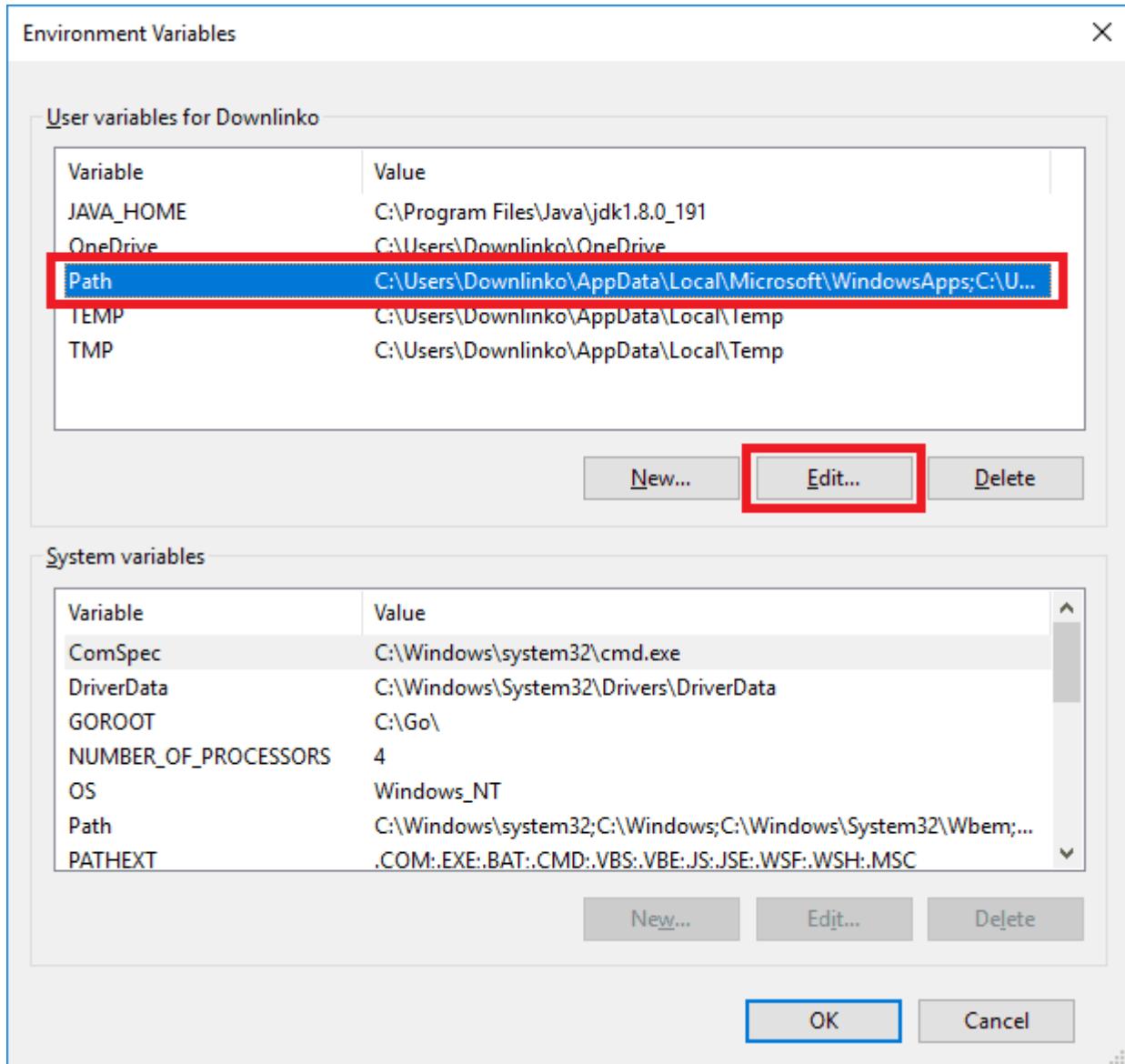
In this tutorial, the Java installation directory is C:\Program Files\Java\jdk1.8.0_191.

Click **OK**.



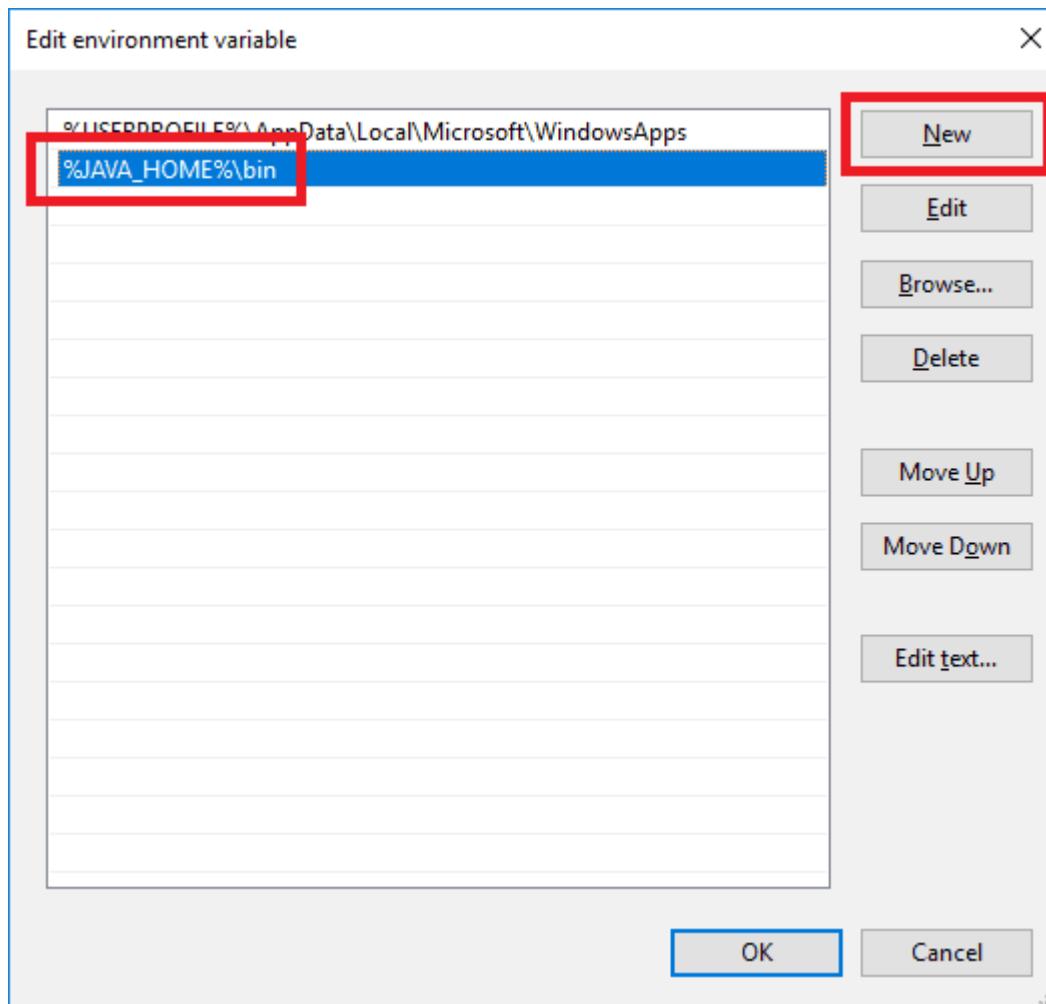
Next, we need to configure the PATH environment variable so we can run Java from a command prompt.

Select the **Path** variable. Click on **Edit...**.

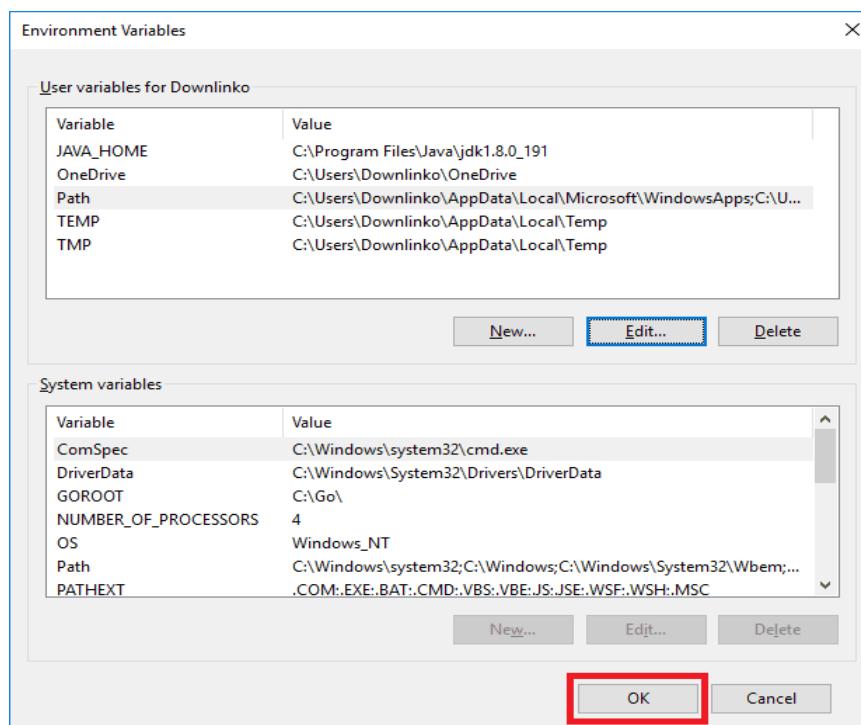


Click on **New...** and type “%JAVA_HOME%\bin” as shown below.

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.

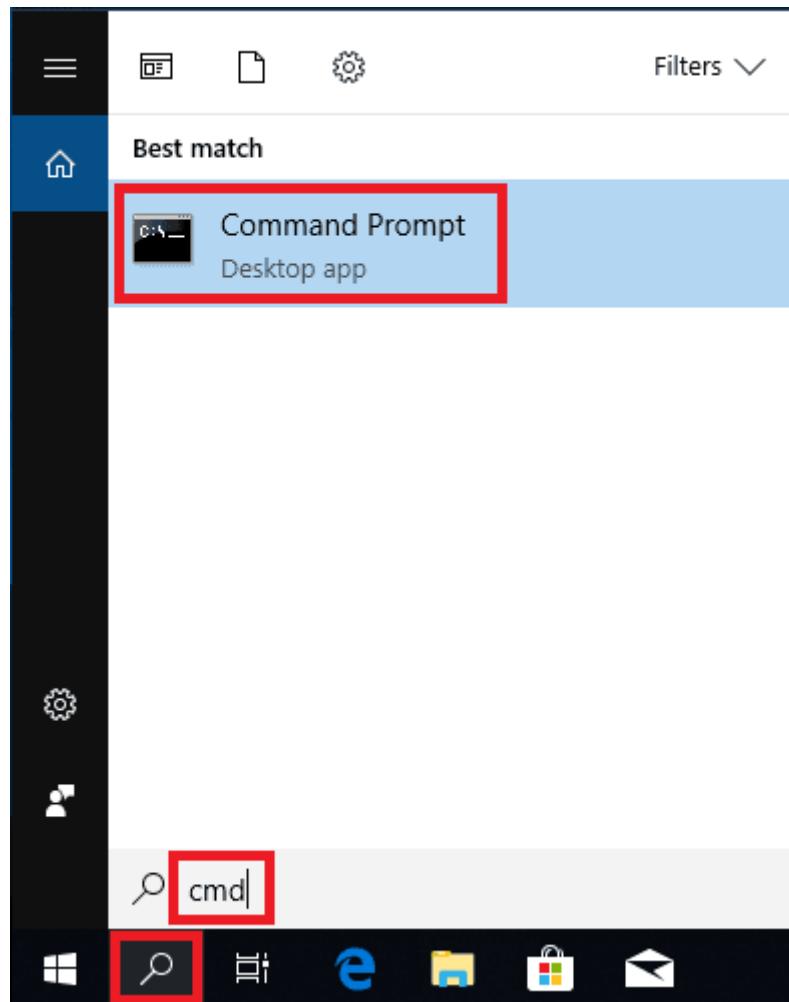
On Windows 7 you cannot add extra values for an existing **Path** variable. You need to append “**%JAVA_HOME%\bin**” at the end of the variable value instead.

Test

Let's test the setup.

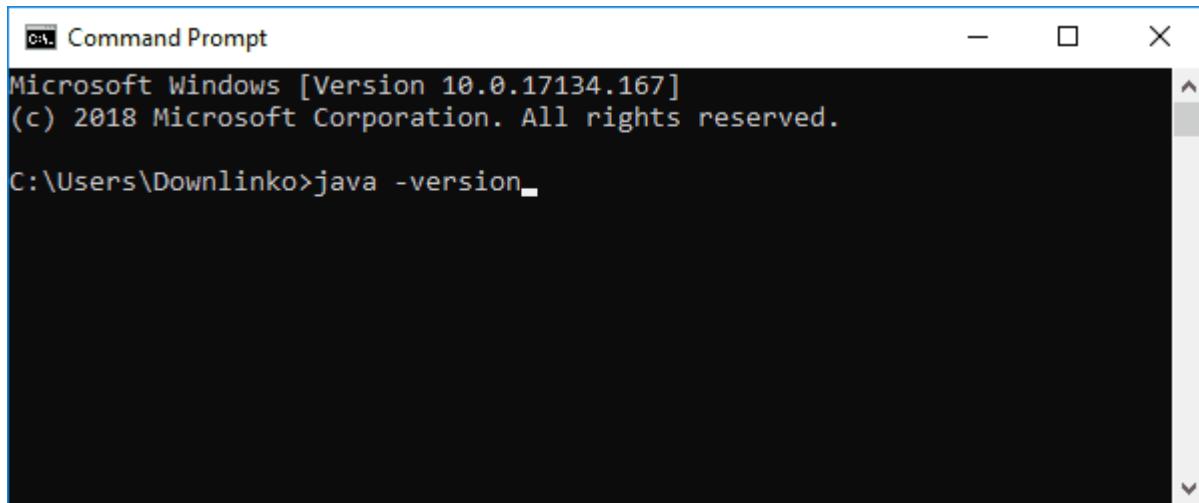
Click on the search button. Then type “**cmd**” (without quotes).

Click on the **Command Prompt** shortcut.



Wait for the command prompt to open.

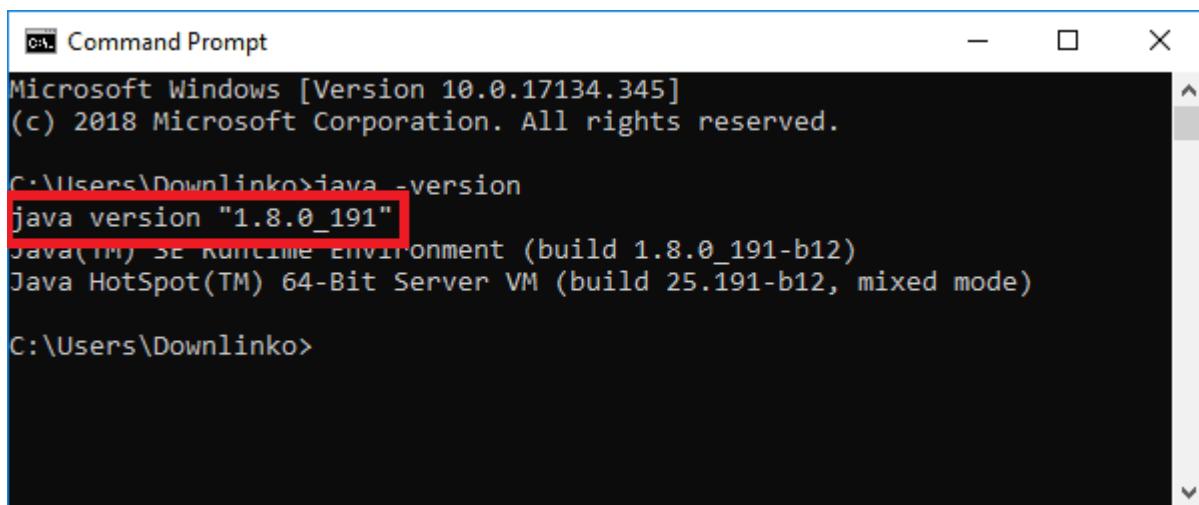
Type “**java -version**” and press **ENTER**.



```
Command Prompt
Microsoft Windows [Version 10.0.17134.167]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Downlinko>java -version
```

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



```
Command Prompt
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Downlinko>java -version
java version "1.8.0_191"
java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)

C:\Users\Downlinko>
```

After completing the Prerequisites Start Installing Eclipse.

STEP 1

Install ABAP Development Tools (ADT)

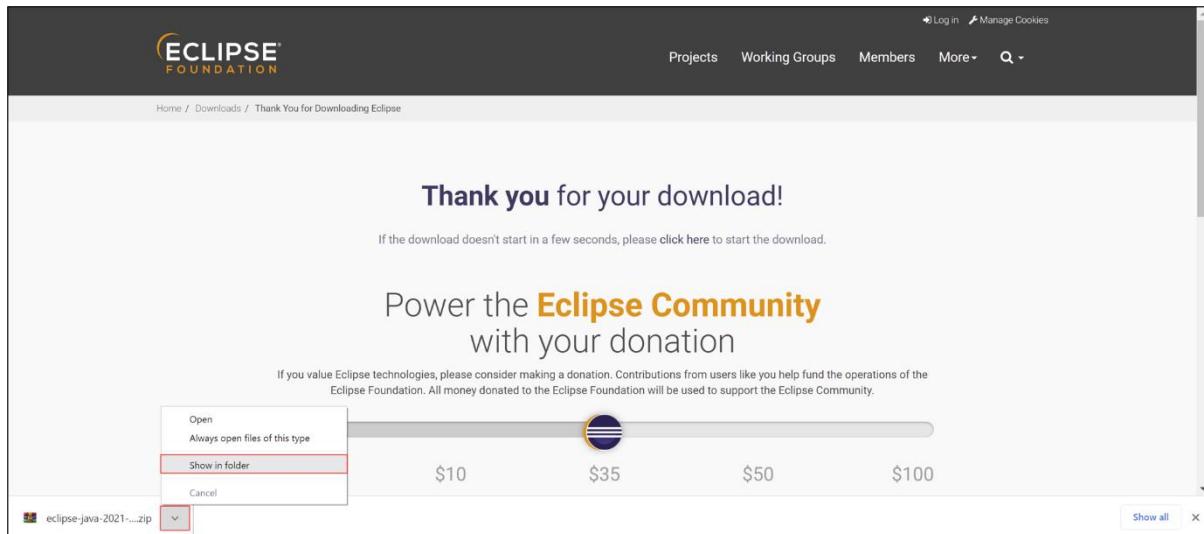
[1] Open the [Eclipse download page](#) to download the corresponding Eclipse version.

The screenshot shows the Eclipse Foundation website's 'Downloads' section for the 'Eclipse IDE 2021-06 R'. It includes links for 'Eclipse Installer', 'Eclipse Packages', and 'Eclipse Developer Builds'. A sidebar for 'Red Hat Developer' is visible. The main content area highlights the 'Eclipse IDE for Java Developers' package, which is 320 MB in size and has 626,504 downloads. A large 'Download' button is present, with a red box drawn around it to indicate the action step.

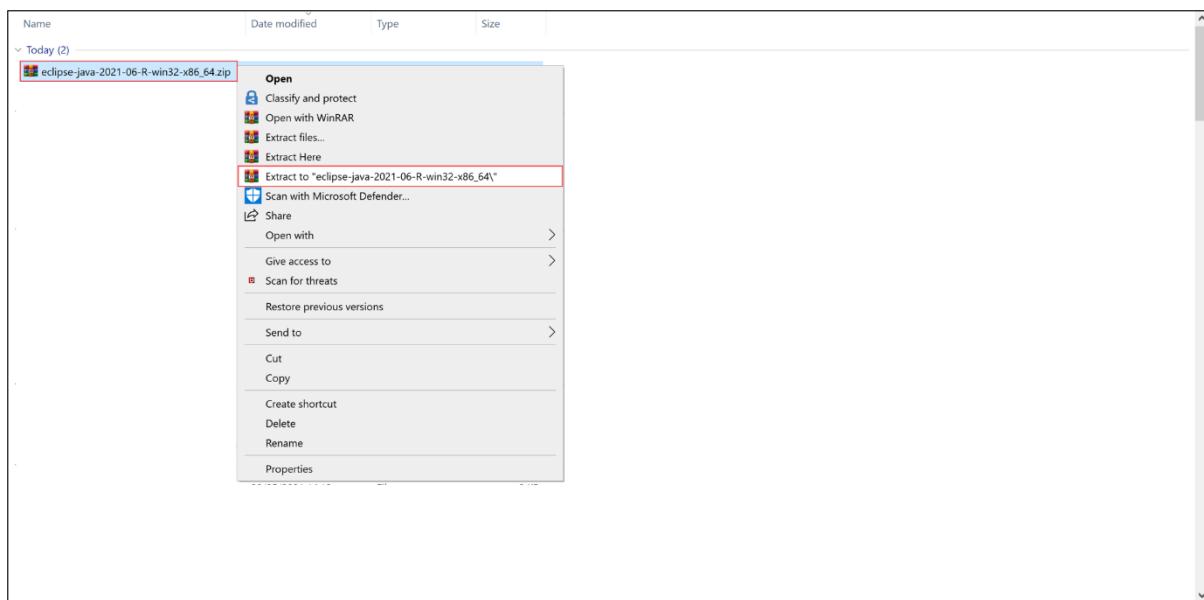
[2] Click Download.

The screenshot shows the 'Eclipse downloads - Select a mirror' page. It displays a large 'Download' button and a 'SHA-512' hash value for the file 'eclipse-java-2021-06-R-win32-x86_64.zip'. A red box highlights the 'SHA-512' link. To the right, there are sections for 'Other options for this file' (including 'All mirrors (xml)' and 'Direct link to file') and 'Related Links' (including 'Donate', 'Becoming a mirror site', 'Updating and installing Eclipse components', and 'Eclipse forums').

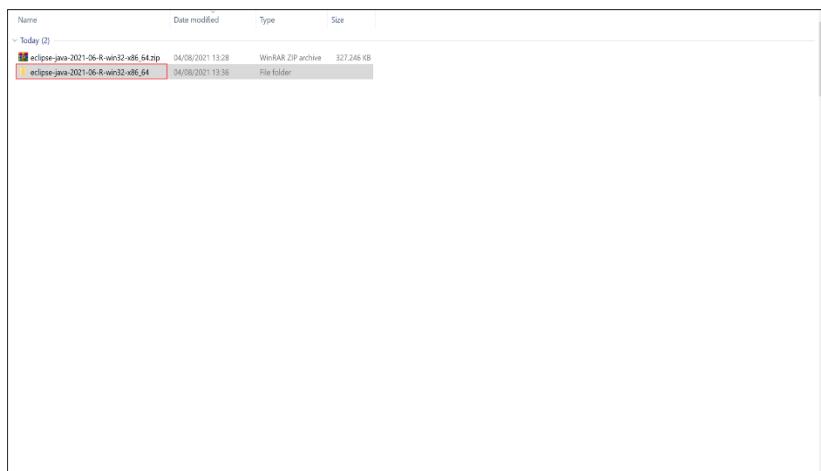
[3] Select Show in folder in your browser.



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



[5] Open the Eclipse-Java folder.



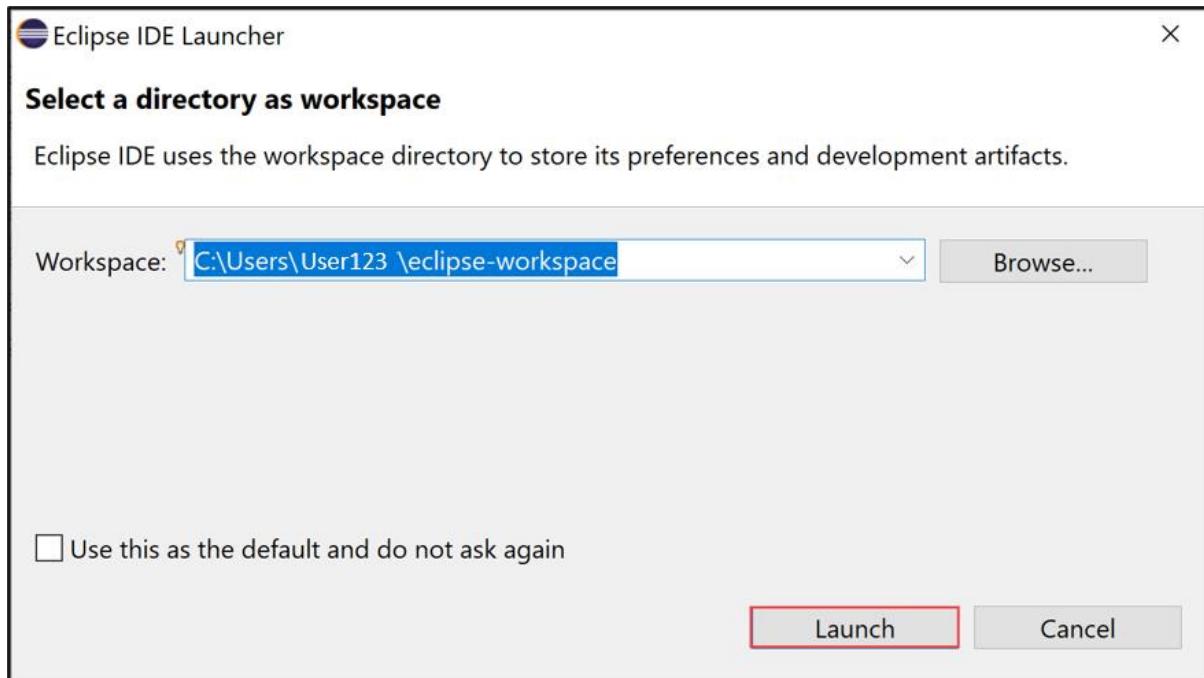
[6] Open the Eclipse folder.

| Name | Date modified | Type | Size |
|---------|------------------|-------------|------|
| eclipse | 12/06/2021 20:32 | File folder | |

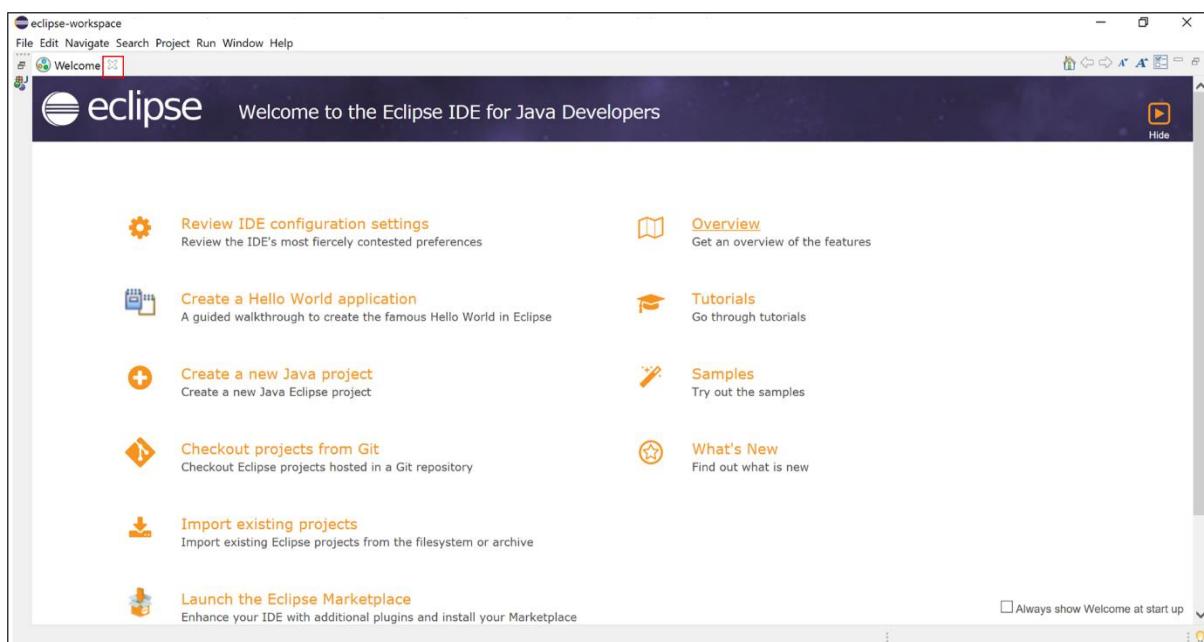
[7] Double-click `eclipse.exe` to run the application.

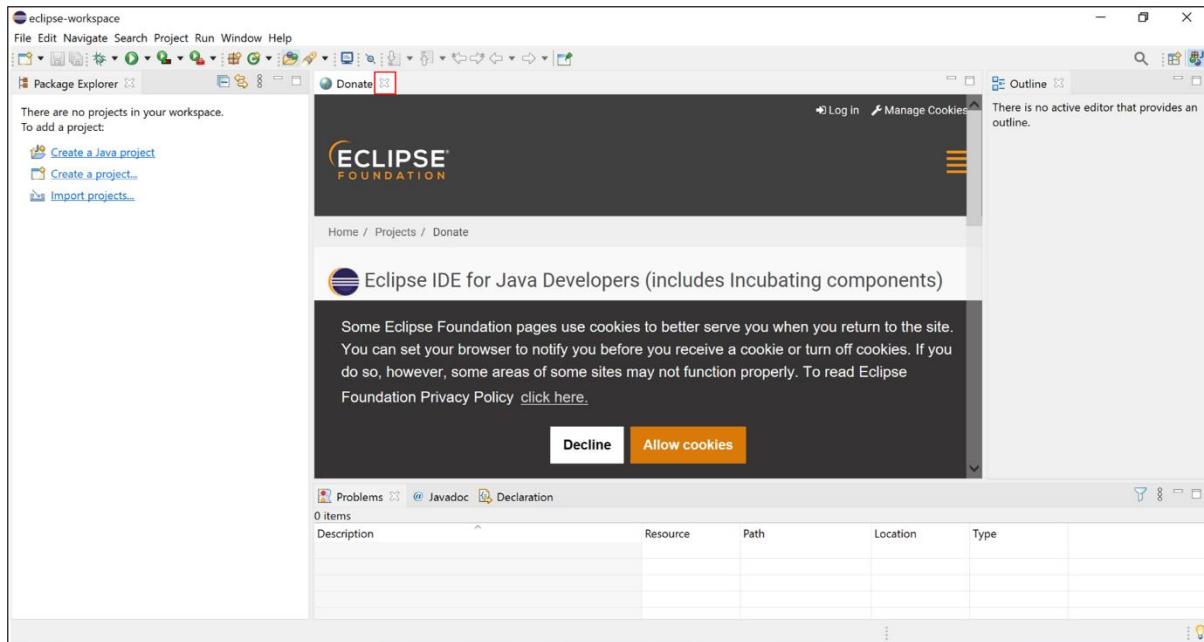
| Name | Date modified | Type | Size |
|-----------------|------------------|-----------------------|--------|
| configuration | 12/06/2021 20:31 | File folder | |
| dropins | 12/06/2021 20:31 | File folder | |
| features | 12/06/2021 20:31 | File folder | |
| p2 | 12/06/2021 20:31 | File folder | |
| plugins | 12/06/2021 20:31 | File folder | |
| readme | 12/06/2021 20:31 | File folder | |
| .eclipseproduct | 11/06/2021 20:06 | ECLIPSEPRODUCT... | 1 KB |
| artifacts.xml | 12/06/2021 20:31 | XML Document | 115 KB |
| eclipse.exe | 12/06/2021 20:32 | Application | 417 KB |
| eclipse.ini | 12/06/2021 20:31 | Configuration sett... | 1 KB |
| eclipsec.exe | 12/06/2021 20:32 | Application | 129 KB |

[8] Launch your workspace.

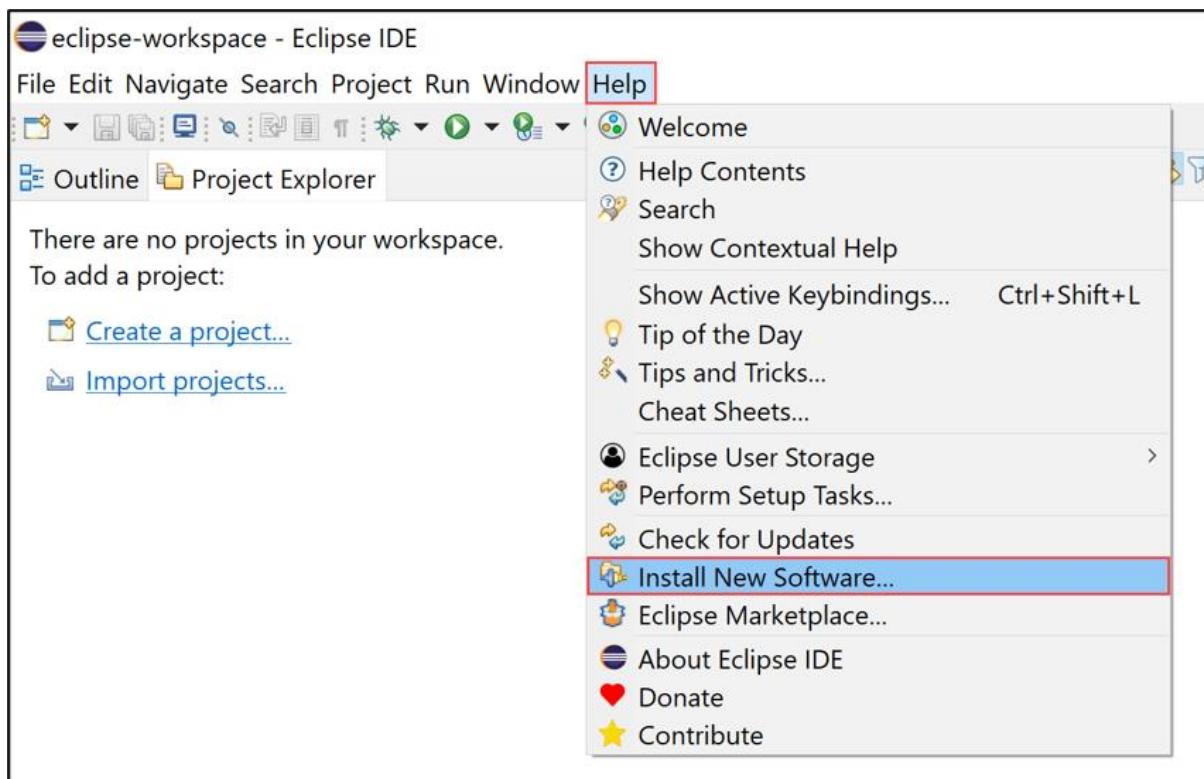


[9] Close both pages.

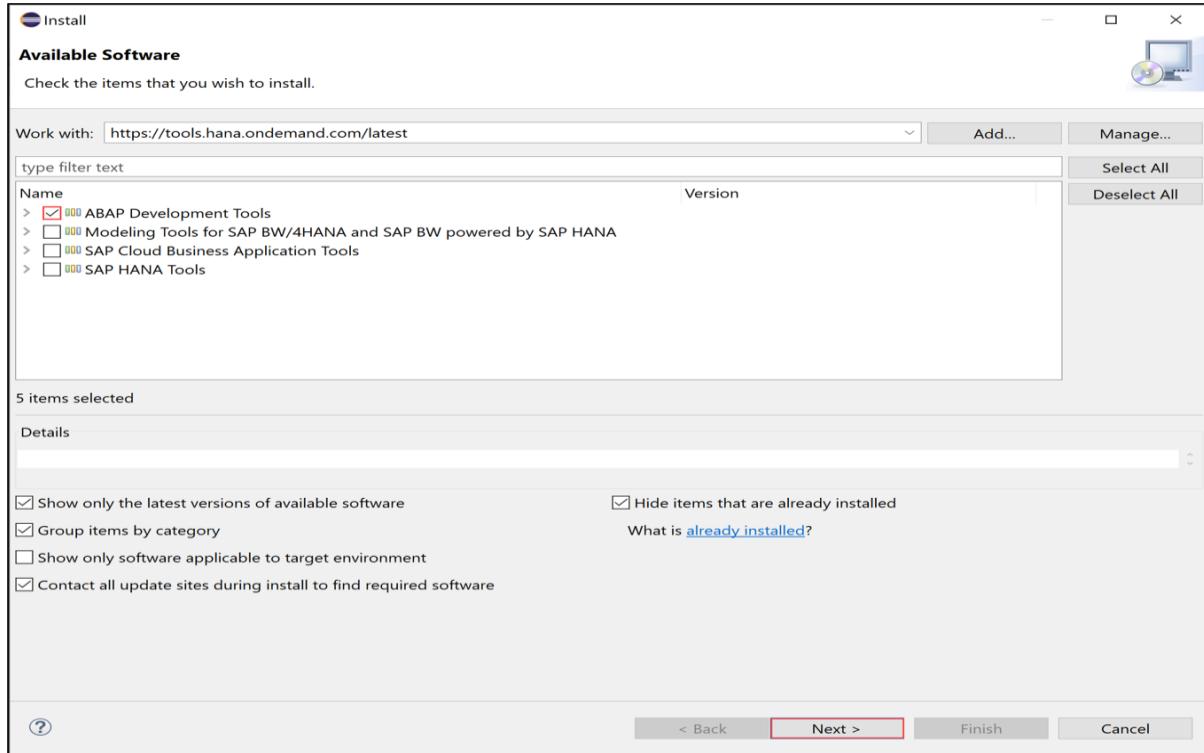




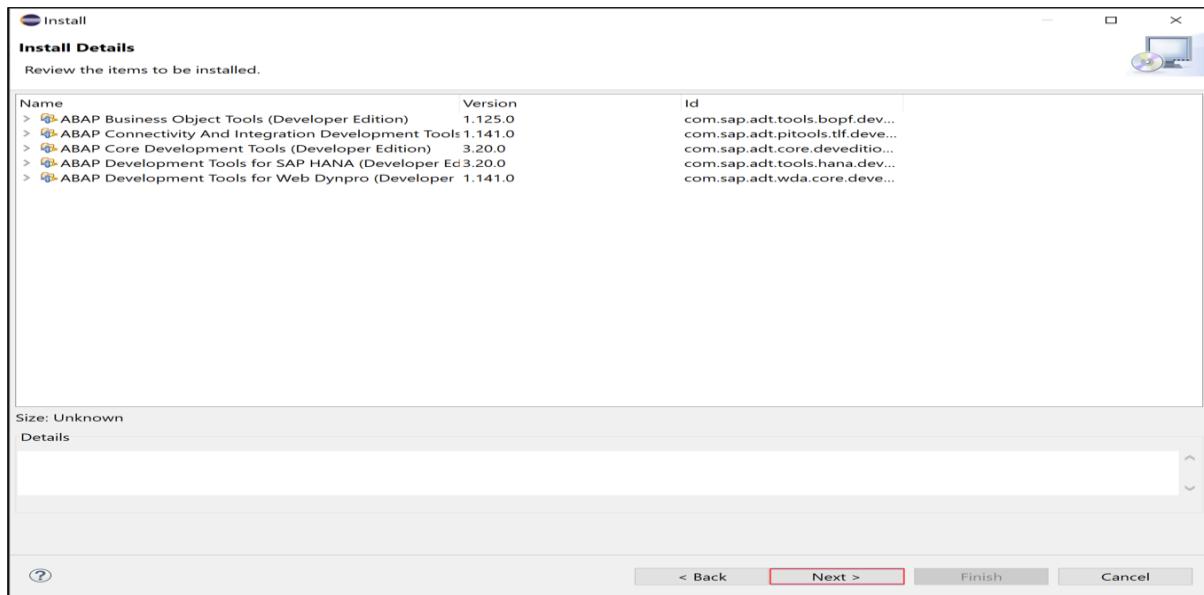
[10] Select Help > Install New Software.



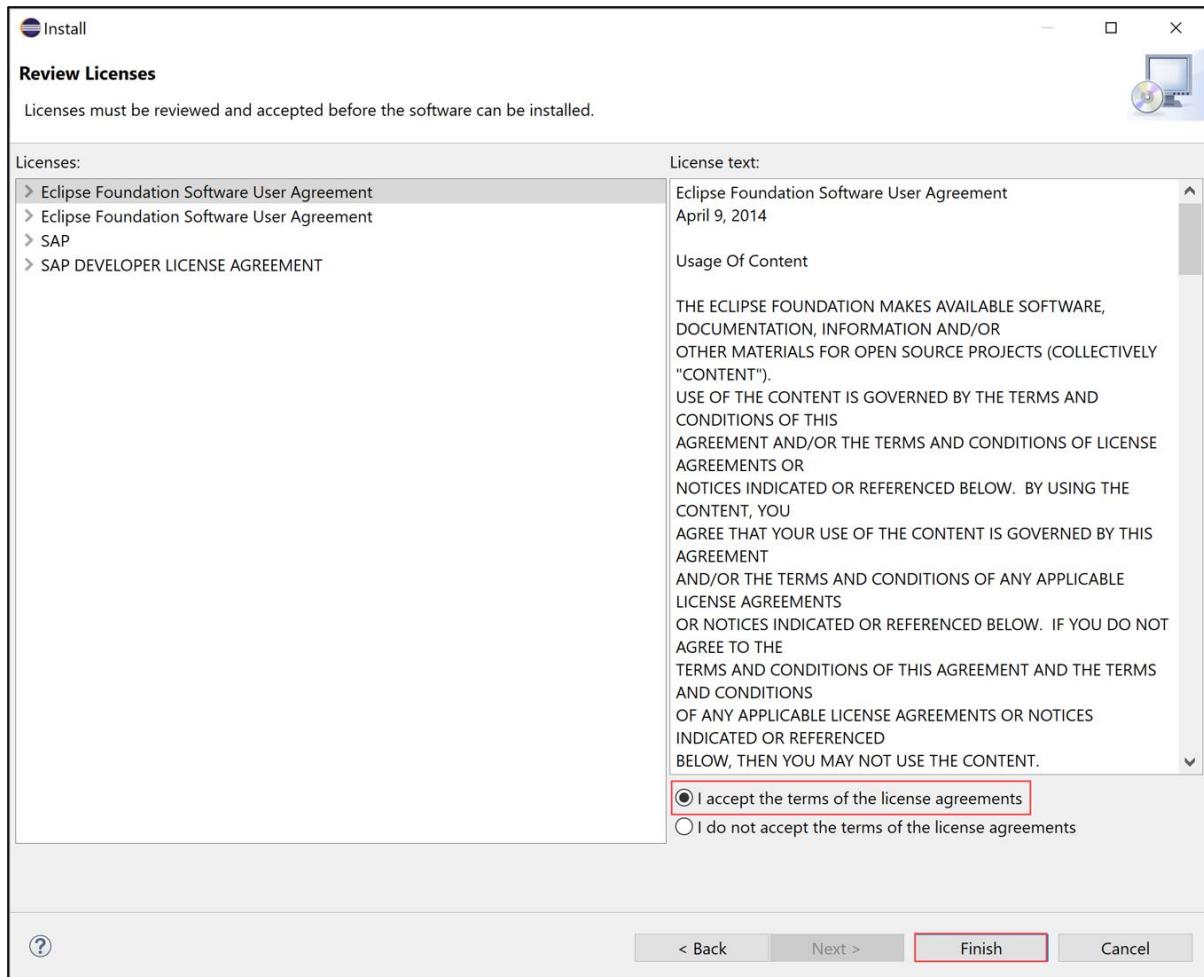
[11] Enter the latest ADT URL <https://tools.hana.ondemand.com/latest> in the Work with section, press enter, select ABAP Development Tools, and click Next >.



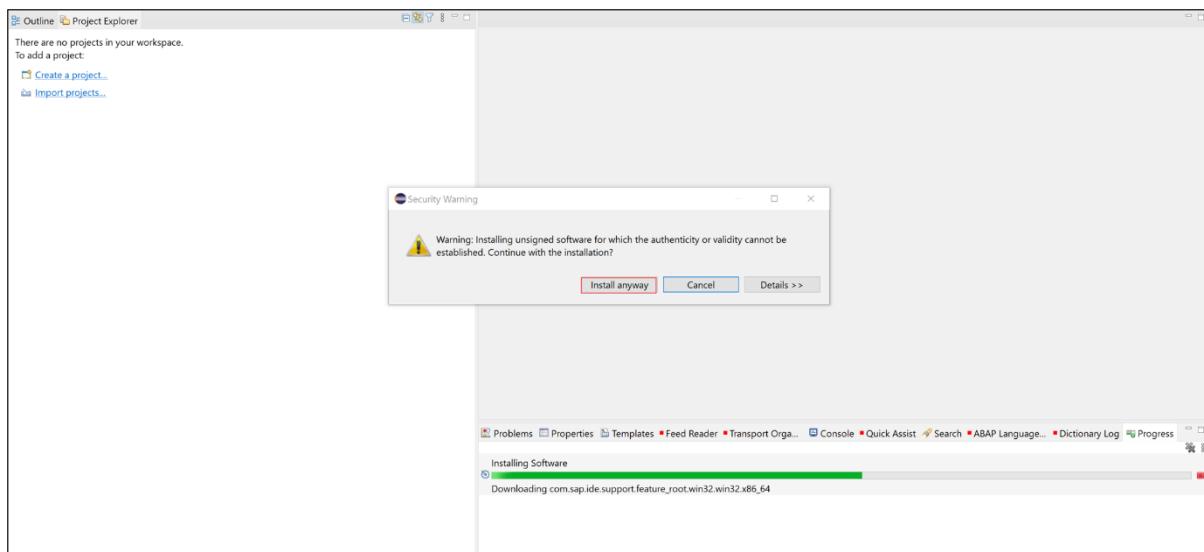
[12] Click Next >.



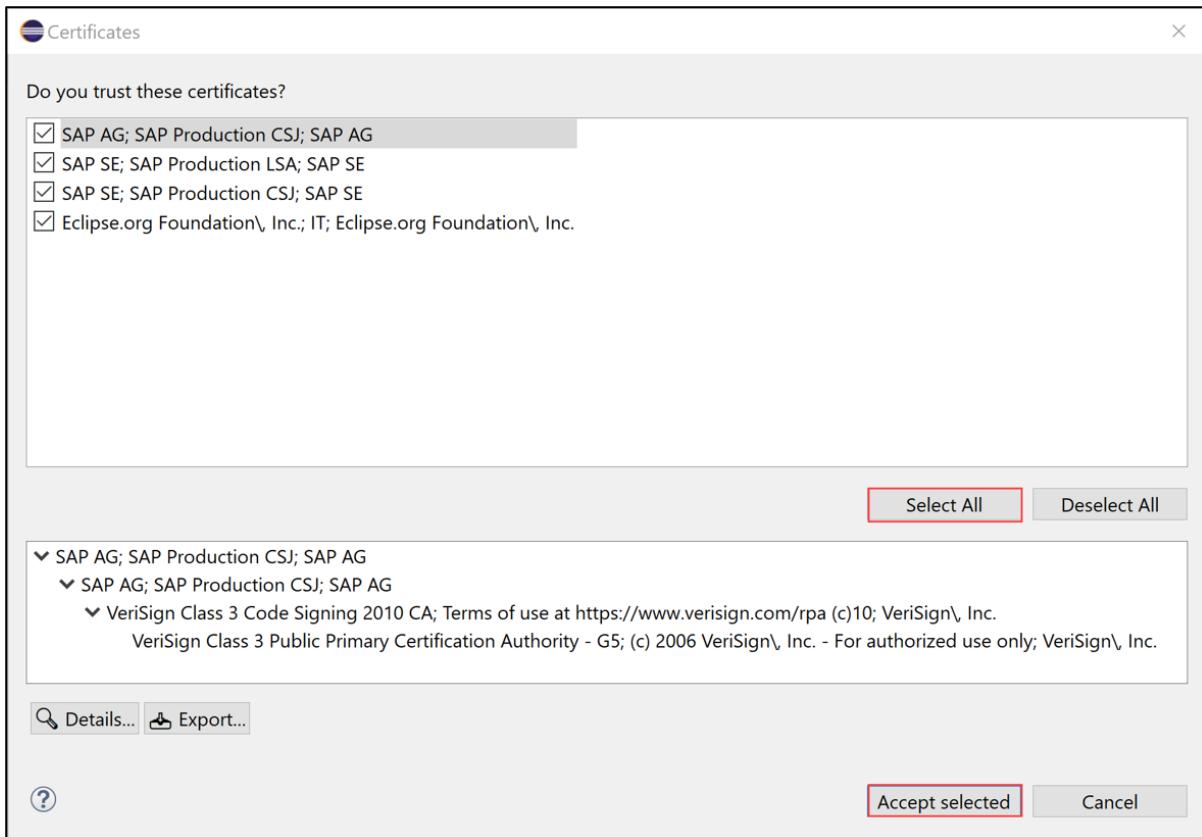
[13] Accept the license agreement and click Finish.



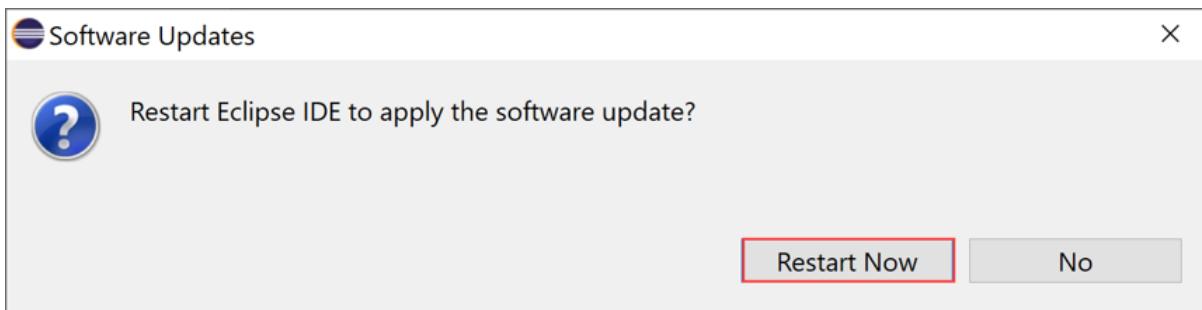
[14] Now ADT will be installed. Select Install anyway.



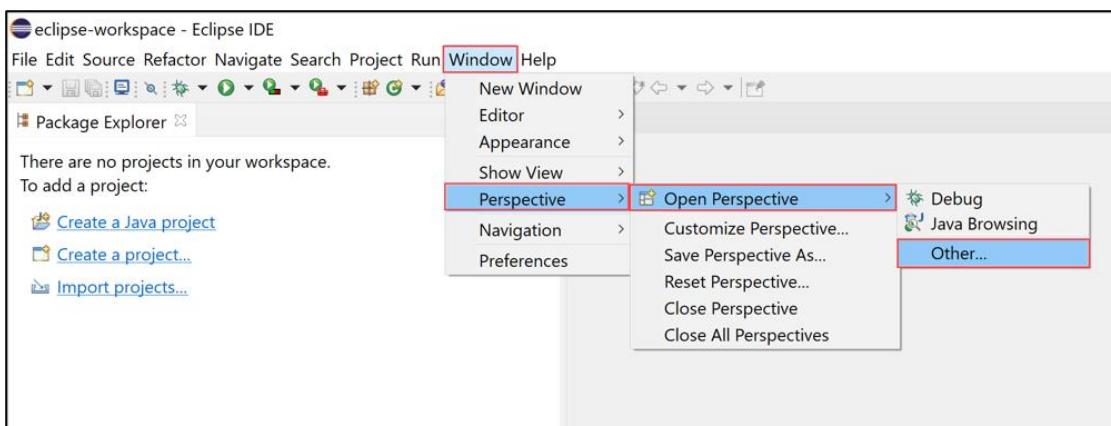
[15] Click Select All and Accept selected.



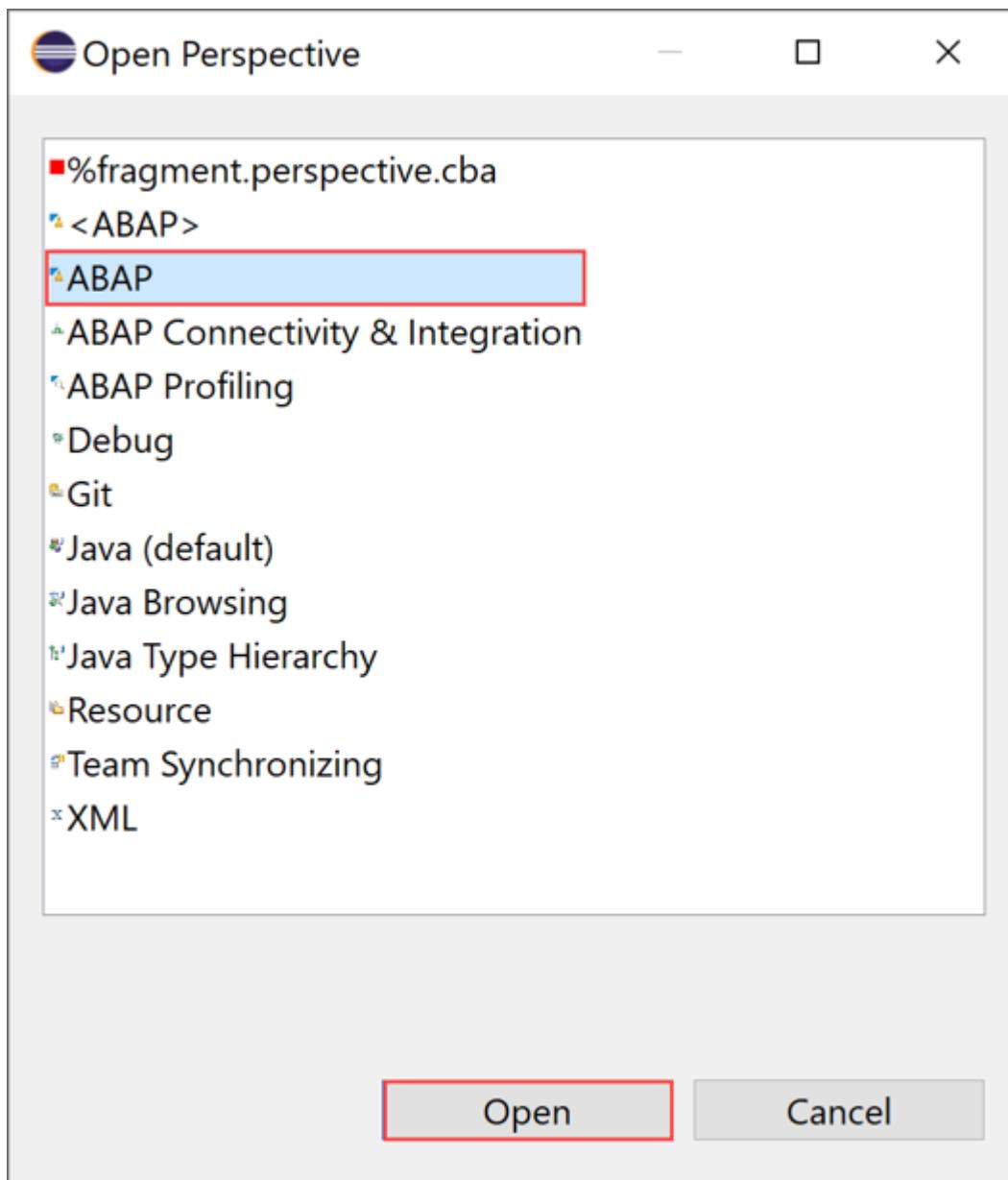
[16] Click Restart Now.



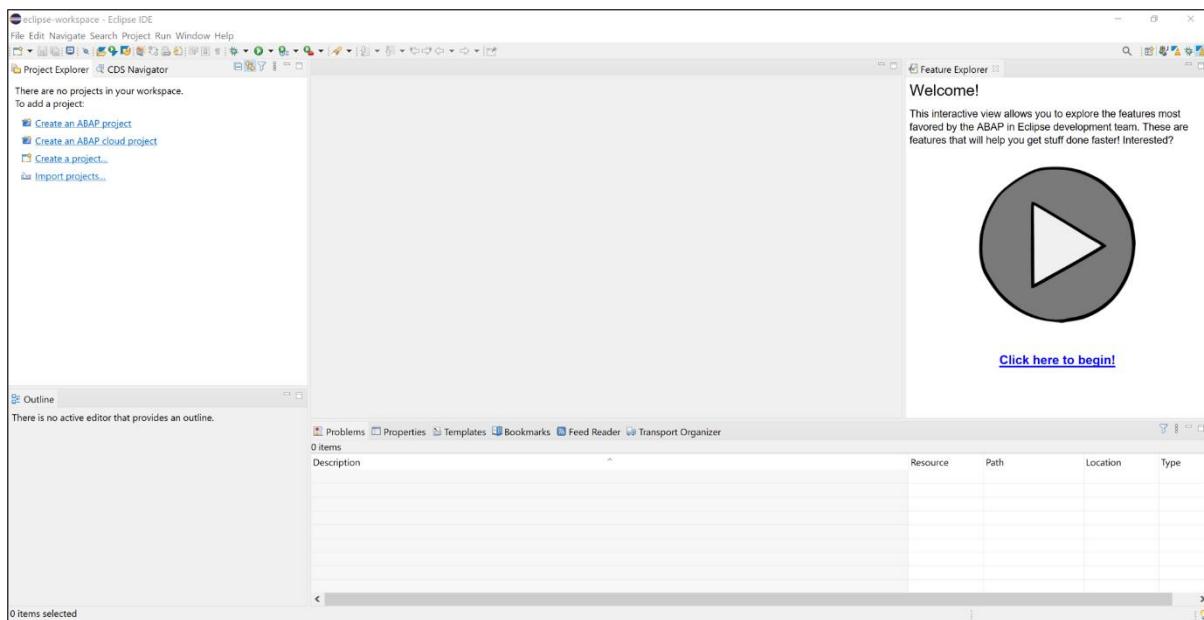
[17] Now ADT is installed. Switch to the ABAP perspective. Therefore select Window > Perspective > Other Perspective > Other.



[18] Then select ABAP and click Open.



[19] Check your result.



STEP 2

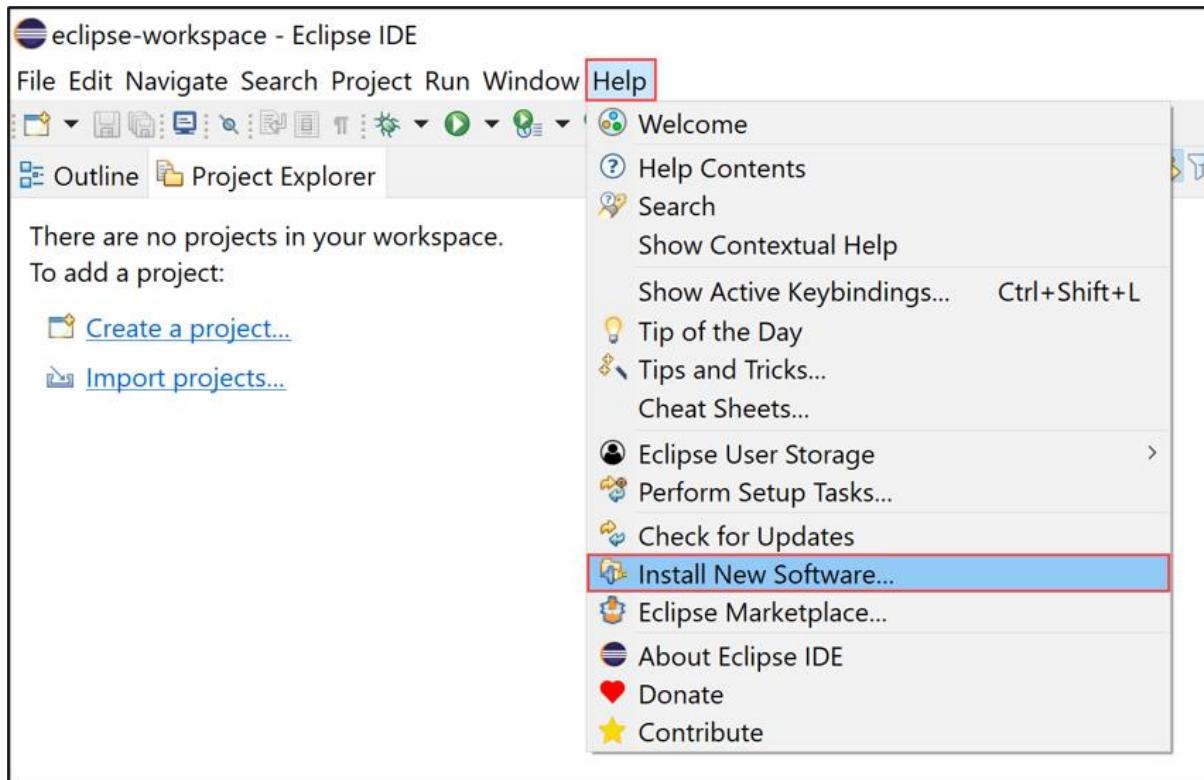
Install abapGit plugging:

HINT:

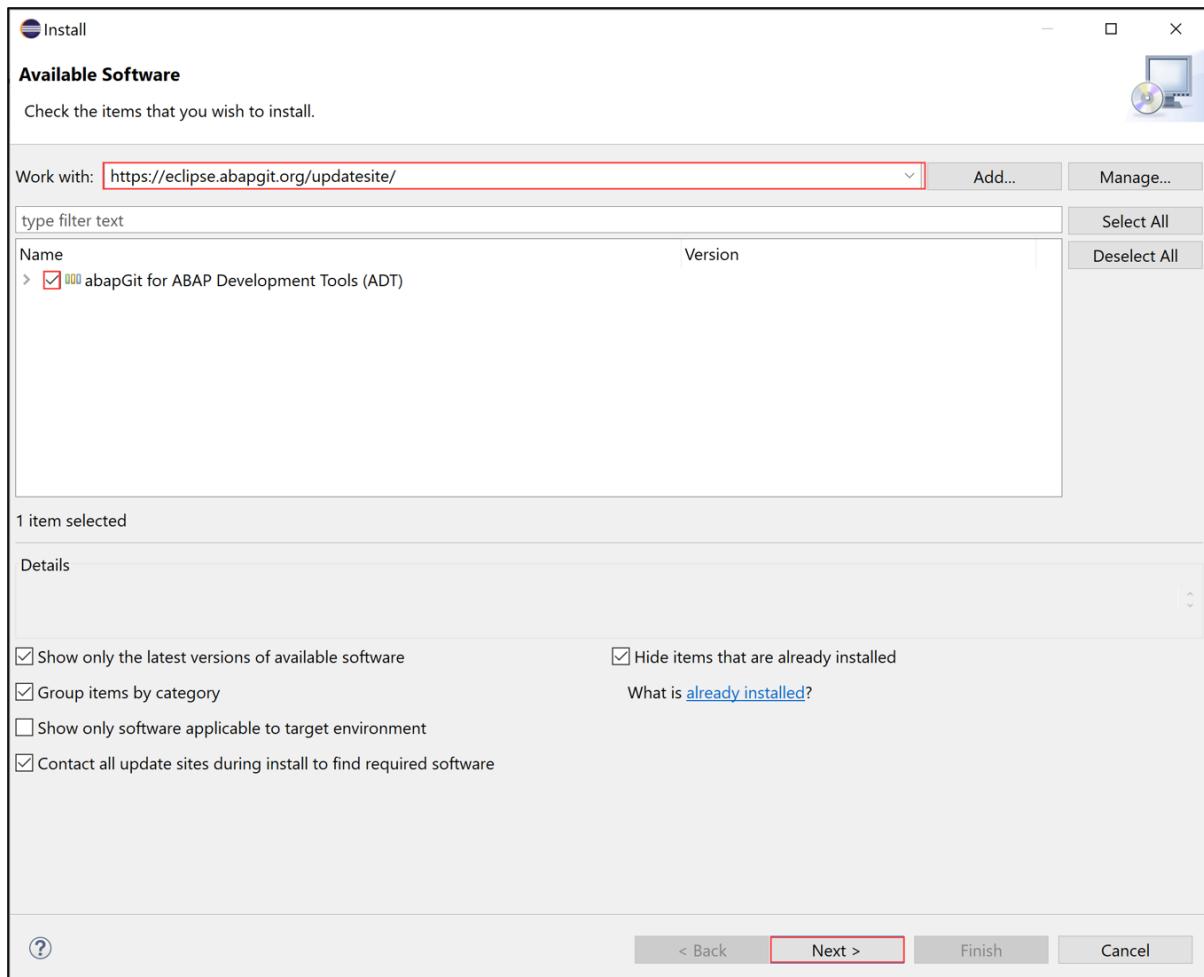
Step 2 is only mandatory for cloud users.

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

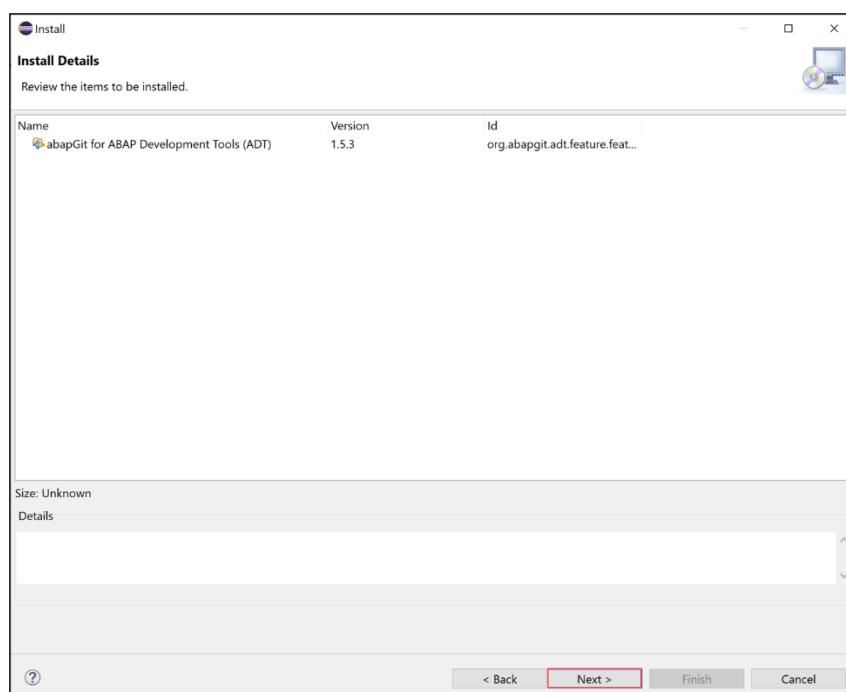
- [1] Open Eclipse and select Help > Install New Software.



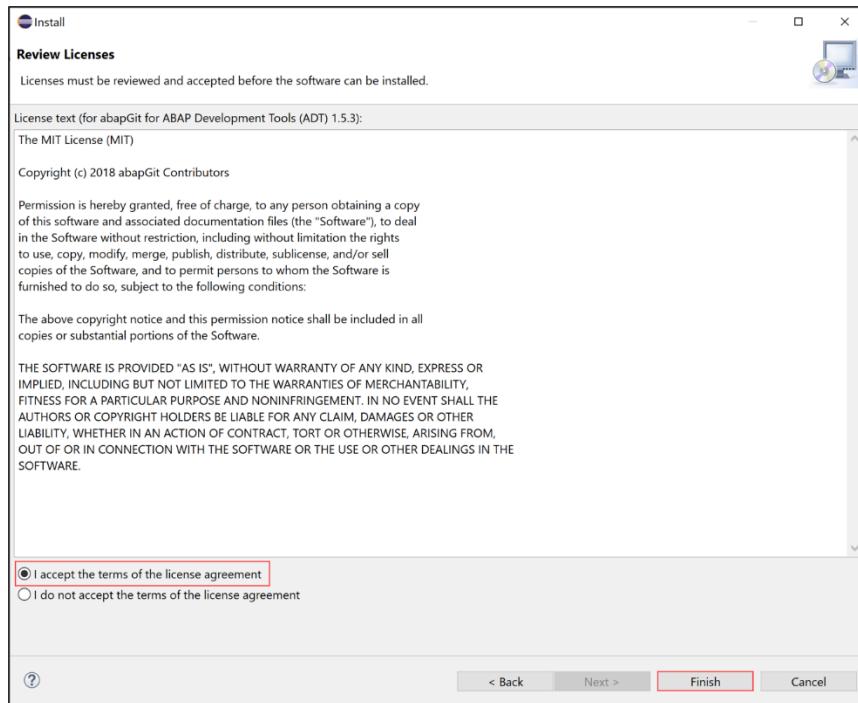
[2] Enter the abapGit URL <https://eclipse.abapgit.org/updatesite/> in the Work with section, press enter, select abapGit for ABAP Development Tools (ADT) and click Next >.



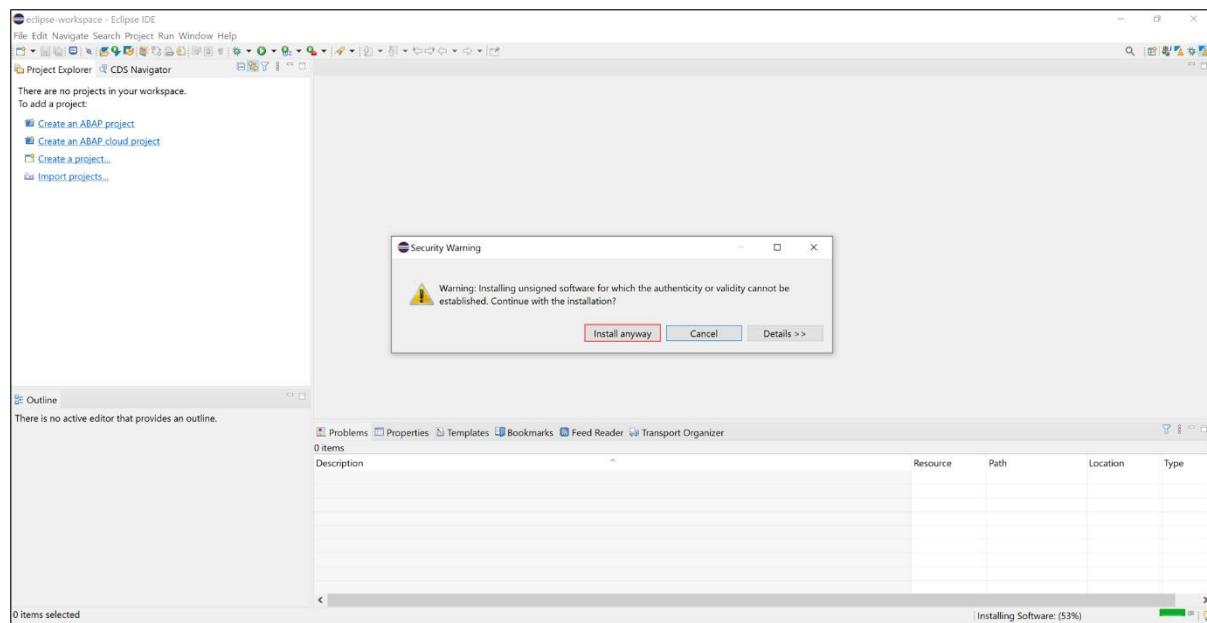
[3] Click Next >.

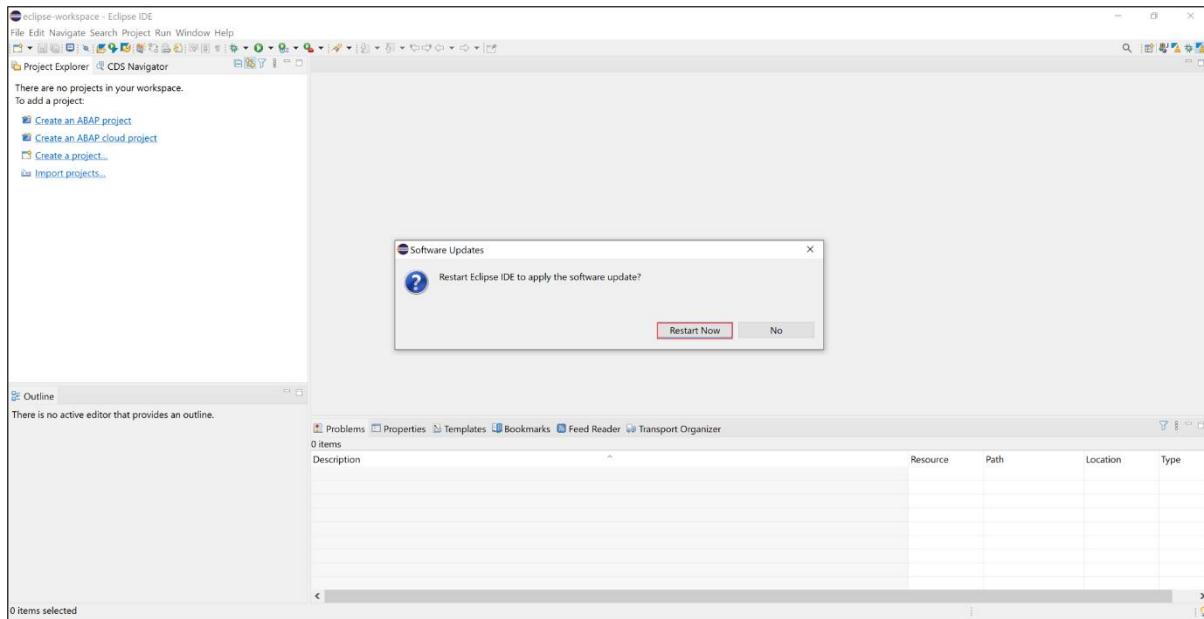
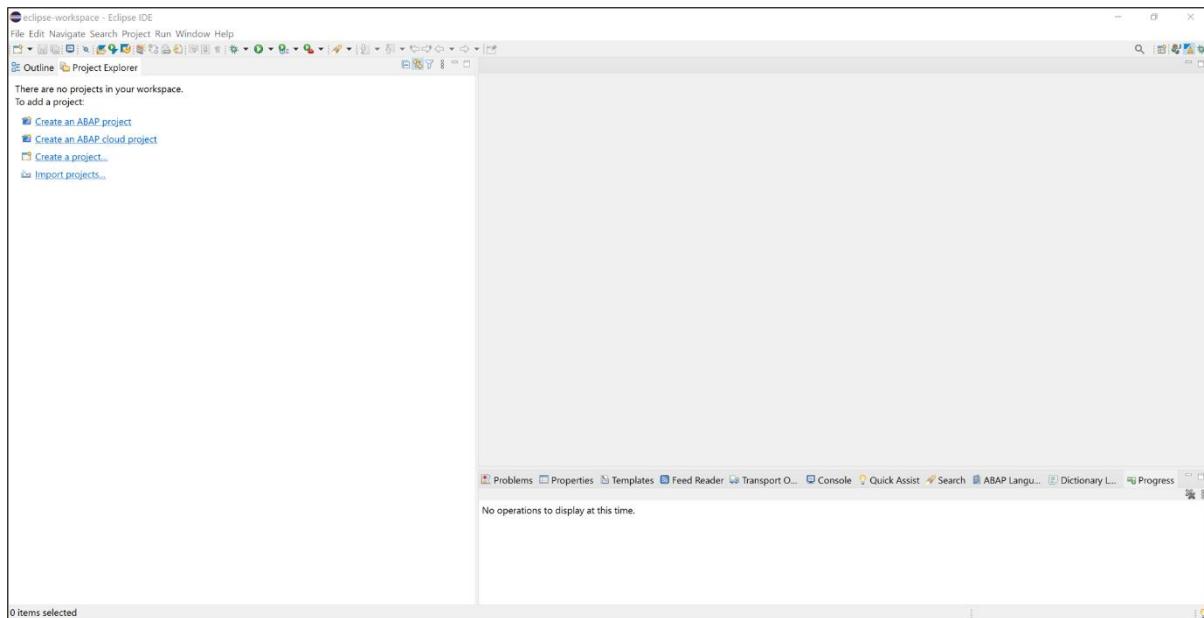


[4] Accept the license agreement and click Finish.



[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`
`@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).

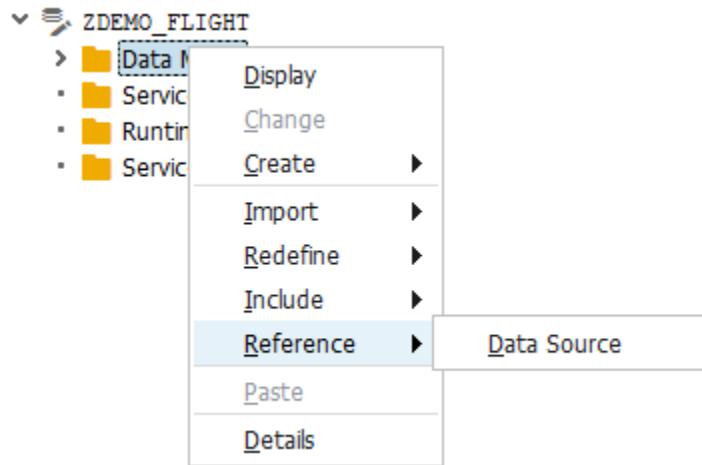
I've tried to place the chart (and respective annotations) in a single view, but the Smart Chart wasn't rendered properly by the UI5 application. You can also notice that we define the association as `[0..*]` instead of `[0..1]`, if you don't follow this convention the chart will not appear in the screen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

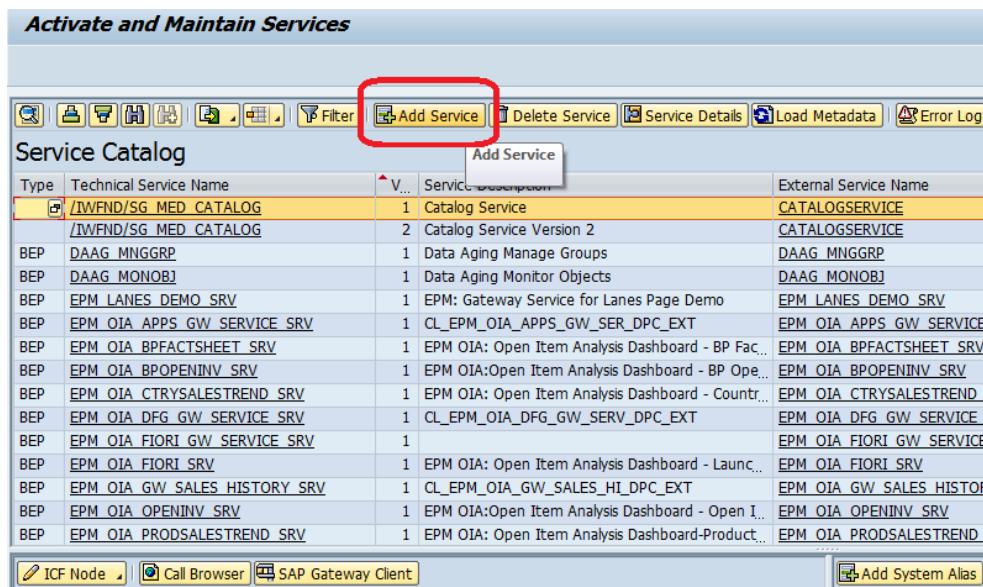
If you do not know how to create a service, please follow these steps. (Otherwise Skip this step)

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.

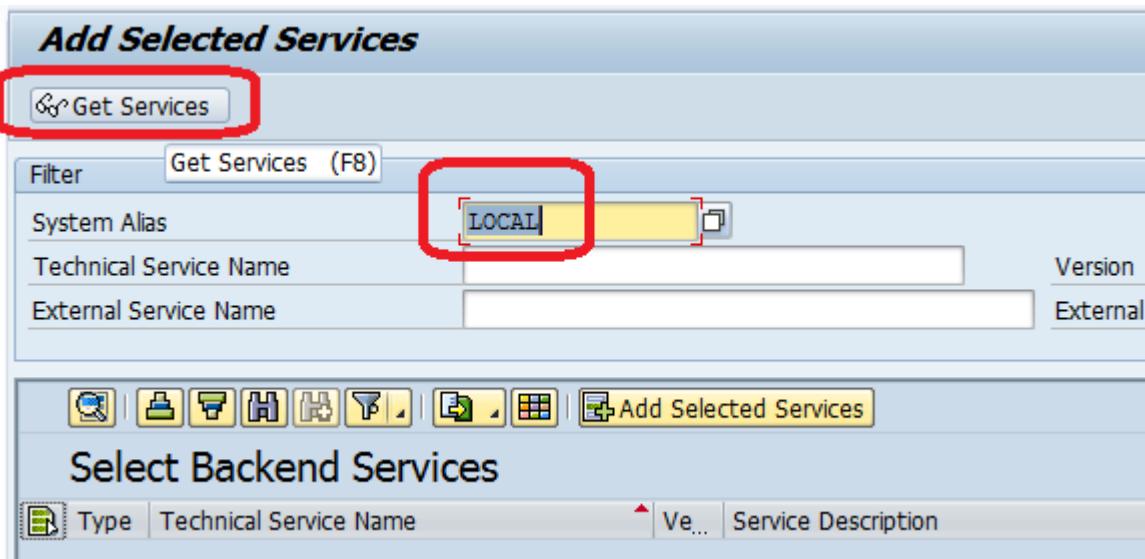
How to Register an ODATA Service?

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’.

The screenshot shows the SAP Fiori Launchpad interface. At the top, there are input fields for 'System Alias' (LOCAL), 'Technical Service Name', 'Version', and 'External Service Name'. Below these is a toolbar with various icons. A red box highlights the 'Add Selected Services' button in the toolbar. The main area is titled 'Select Backend Services' and contains a table with columns: Type, Technical Service Name, Version, Service, and External Service Name. A red box highlights the row for 'BEP ZTEST_ODATA_SRV'. The 'Service' column for this row shows the value '1 Test Odata service'.

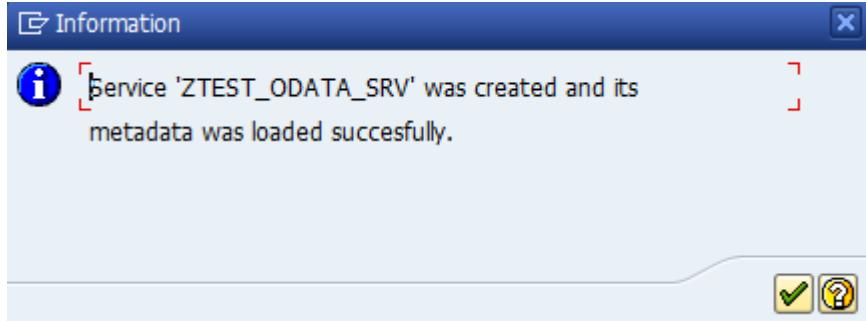
9. In appear pop-up click select input details as shown in below scree and here enters your desired package, for example purpose \$TMP

The screenshot shows the 'Add Service' dialog box. The 'Service' tab is active, displaying the following details:

- Technical Service Name: ZTEST_ODATA_SRV
- Service Version: 1
- Description: Test Odata service
- External Service Name: ZTEST_ODATA_SRV
- Namespace: (empty)
- External Mapping ID: (empty)
- External Data Source Type: C

The 'Model' tab is also visible. A red box highlights the 'Package Assignment' field, which contains '\$TMP'. Other tabs like 'Select Back' and 'ICF Node' are partially visible.

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.

The screenshot shows the SAP Activate and Maintain Services interface. The title bar says 'Activate and Maintain Services'. Below it is a toolbar with various icons. The main area is titled 'Service Catalog' and contains a table of services. One row in the table is highlighted in yellow, corresponding to the service created in step 10. The columns in the table include Type, Technical Service Name, Version, Service Description, External Service Name, Namespace, OAU, Soft State Status, and Is SAP Service. The highlighted row for 'ZTEST_ODATA_SRV' has 'Not Supported' checked in the Is SAP Service column. Other services listed include CL_SRA020_PO_TRACKING_DPC_EXT, QUICKVIEW, PAGEBUILDER SERVICE V 0_1, PAGE_BUILDER_PERS, PAGE_BUILDER_CUST, PAGE_BUILDER_CONF, LAUNCHPAD, INTEROP, TERM_TERMINOLOGY_SEARCH_SRV, TERM_SUGGEST_SRV, TERM_SEARCH_SRV, and SEPM_HANA_EXT_PA_ODATA_SRV. At the bottom of the screen, there are tabs for ICF Nodes, Call Browser, SAP Gateway Client, and System Aliases. The SAP Gateway Client tab is selected. The ICF Nodes tab shows a single entry for ODATA with a session timeout of 00:00:00 and Standard Mode. The System Aliases tab shows a single entry for LOCAL with Local System Alias checked.

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

SAP Gateway Client

HTTP Request

Header Name Value

HTTP Response - Processing Time = 3242 ms

| Header Name | Value |
|----------------|-------|
| -status_code | 200 |
| -status_reason | OK |

```

<?xml version='1.0' encoding='utf-8'?>
- <app:service xml:lang='en' xml:base='http://          0/sap/opu/odata/sap/ZTEST_ODATA_SRV/
  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:pageable='false' sap:addressable='false' sap:content-version='1' href='MaterialListSet'>
    <atom:title type='text'>MaterialListSet</atom:title>
    <app:member-type>MaterialList</app:member-type>
  </app:collection>
</app:workspace>
<atom:link rel='self' href='http://          /sap/opu/odata/sap/ZTEST_ODATA_SRV/*'>
<atom:link rel='latest-version'
  href='http://          /sap/opu/odata/sap/ZTEST_ODATA_SRV/'>
</app:service>
```

SAP Gateway Client

HTTP Request

Header Name Value

HTTP Response - Processing Time = 453 ms

| Header Name | Value |
|----------------|-------|
| -status_code | 200 |
| -status_reason | OK |

```

- <edmx:DataServices m:DataServiceVersion='2.0'>
- <Schema Namespace='ZTEST_ODATA_SRV' xml:lang='en' sap:schema-version='1'
  xmlns='http://schemas.microsoft.com/ado/2008/09/edm'>
- <EntityType Name='MaterialList' sap:content-version='1'>
  - <Key>
    <PropertyRef Name='MATERIAL' />
  </Key>
  <Property Name='MATERIAL' Type='Edm.String' Nullable='false' sap:label='MATERIAL
  NUMBER' sap:creatable='false' sap:updatable='false' sap:sortable='false'
  sap:filterable='false' />
  <Property Name='MYP' Type='Edm.String' Nullable='false' sap:label='MATERIAL
  TYPE' sap:creatable='false' sap:updatable='false' sap:sortable='false'
  sap:filterable='false' />
</EntityType>
- <EntityContainer Name='ZTEST_ODATA_SRV_Entities' m:IsDefaultEntityContainer='true'
  sap:supported-formats='atom json xslx'>
  <EntitySet Name='MaterialListSet' EntityType='ZTEST_ODATA_SRV.MaterialList'
    sap:creatable='false' sap:updatable='false' sap:deletable='false'
    sap:pageable='false' sap:addressable='false' sap:content-version='1' />
</EntityContainer>
```

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.

The screenshot shows the SAP Fiori Launchpad interface for creating a new application. The top navigation bar has tabs: 'Template Selection', 'Basic Information', 'Data Connection', 'Annotation Selection', 'Template Customization', and 'Confirmation'. The 'Template Selection' tab is active. Below the tabs, the title 'New List Report Application' is displayed, followed by 'Template Selection'. There are four filter fields: 'Search' (with a magnifying glass icon), 'Category' (set to 'SAP Fiori Elements'), 'Sort By' (set to 'Sort by recently used'), and 'SAPUI5 Version' (set to 'SAP Innovation (1.48)'). Below the filters, three application templates are listed in cards:

- List Report Application**: Represented by a blue card with a list icon and a heart icon.
- Analytical List Page**: Represented by a light gray card with a grid icon and a heart icon.
- Overview Page Application**: Represented by a light gray card with a bar chart icon and a heart icon.

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



New List Report Application

Basic Information

Project Name*

zdemo_flight

App Descriptor Data

Title*

Flights

Namespace

com.au.demo.flights

Description

Flights (Smart Charts)

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.



New List Report Application

Data Connection

Service: ZDEMO_FLIGHT_CDS selected.

Choose a service from one of the sources listed below.

Sources Choose a system, explore, and select a service. Optionally, you can drill down into the service details.

| Service Catalog | | | | | | | | | | | | | | | | | | | | |
|------------------|--|--|---------|-------------|--------|------------------|------------------|---------|--------------|--|--|------------|-------------|--|--------------|-------------------|--|-----------|---------|--|
| Workspace | F4D_020 | | | | | | | | | | | | | | | | | | | |
| File System | Services | <input type="text" value="zdemo"/> Show Details Select | | | | | | | | | | | | | | | | | | |
| Service URL | <table border="1"> <thead> <tr> <th>Service</th> <th>Description</th> <th>System</th> </tr> </thead> <tbody> <tr> <td>ZDEMO_FLIGHT_CDS</td> <td>ZDEMO FLIGHT CDS</td> <td>F4D 020</td> </tr> <tr> <td>ZDEMO_FLIGHT</td> <td></td> <td></td> </tr> <tr> <td>FlightDate</td> <td>Flight Date</td> <td></td> </tr> <tr> <td>ConnectionId</td> <td>Connection Number</td> <td></td> </tr> <tr> <td>CarrierId</td> <td>Airline</td> <td></td> </tr> </tbody> </table> | | Service | Description | System | ZDEMO_FLIGHT_CDS | ZDEMO FLIGHT CDS | F4D 020 | ZDEMO_FLIGHT | | | FlightDate | Flight Date | | ConnectionId | Connection Number | | CarrierId | Airline | |
| Service | Description | System | | | | | | | | | | | | | | | | | | |
| ZDEMO_FLIGHT_CDS | ZDEMO FLIGHT CDS | F4D 020 | | | | | | | | | | | | | | | | | | |
| ZDEMO_FLIGHT | | | | | | | | | | | | | | | | | | | | |
| FlightDate | Flight Date | | | | | | | | | | | | | | | | | | | |
| ConnectionId | Connection Number | | | | | | | | | | | | | | | | | | | |
| CarrierId | Airline | | | | | | | | | | | | | | | | | | | |

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.



New List Report Application

Annotation Selection

Select the desired annotation files and rank them in the order in which they will be loaded.

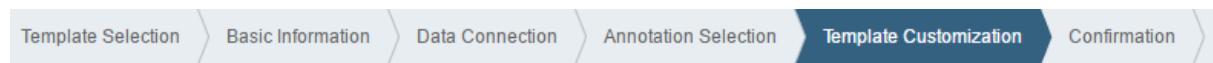
Note: If the annotation files overlap, the one loaded last will overwrite the previous ones.

The selected service contains annotation data.

+ Add Annotation Files

| | Rank | Name | Source |
|-------------------------------------|------|----------------------|--------|
| <input checked="" type="checkbox"/> | 1 | ZDEMO_FLIGHT_CDS_VAN | Remote |

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



New List Report Application

Template Customization

Data Binding

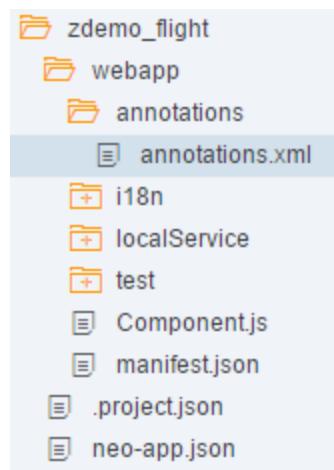
OData Collection*

ZDEMO_FLIGHT

OData Navigation

OData navigation attribute to a collection of items

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.

Annotation Structure

OData Data Source: mainService

OData Entity Type: ZDEMO_FLIGHTType

Search



| Node | Edit Qualifier | Key Information | Expression Type | Value |
|----------------------|----------------|---------------------|-------------------|------------------------------|
| Local Annotations | | | | |
| Annotations | | Source: /zdemo... | | |
| UI.Facets | | | | |
| UI.ReferenceFacet | | ID: PlaneDetails... | | |
| Target * | | AnnotationPath | Navigation | @UI.FieldGroup#FlightDetails |
| Label | | String (i18n) | Flight | |
| ID | | String | PlaneDetailsGroup | |
| UI.ReferenceFacet | | ID: PlaneTypeC... | | |
| Label | | String (i18n) | Occupied Seats | |
| Target * | | AnnotationPath | to_Chart | @UI.Chart#OccupiedSeats |
| ID | | String | PlaneTypeChart | |
| Property Annotations | | | | |
| External Annotations | | | | |
| Annotations | | Source: /sap/op... | | |
| Property Annotations | | | | |

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.

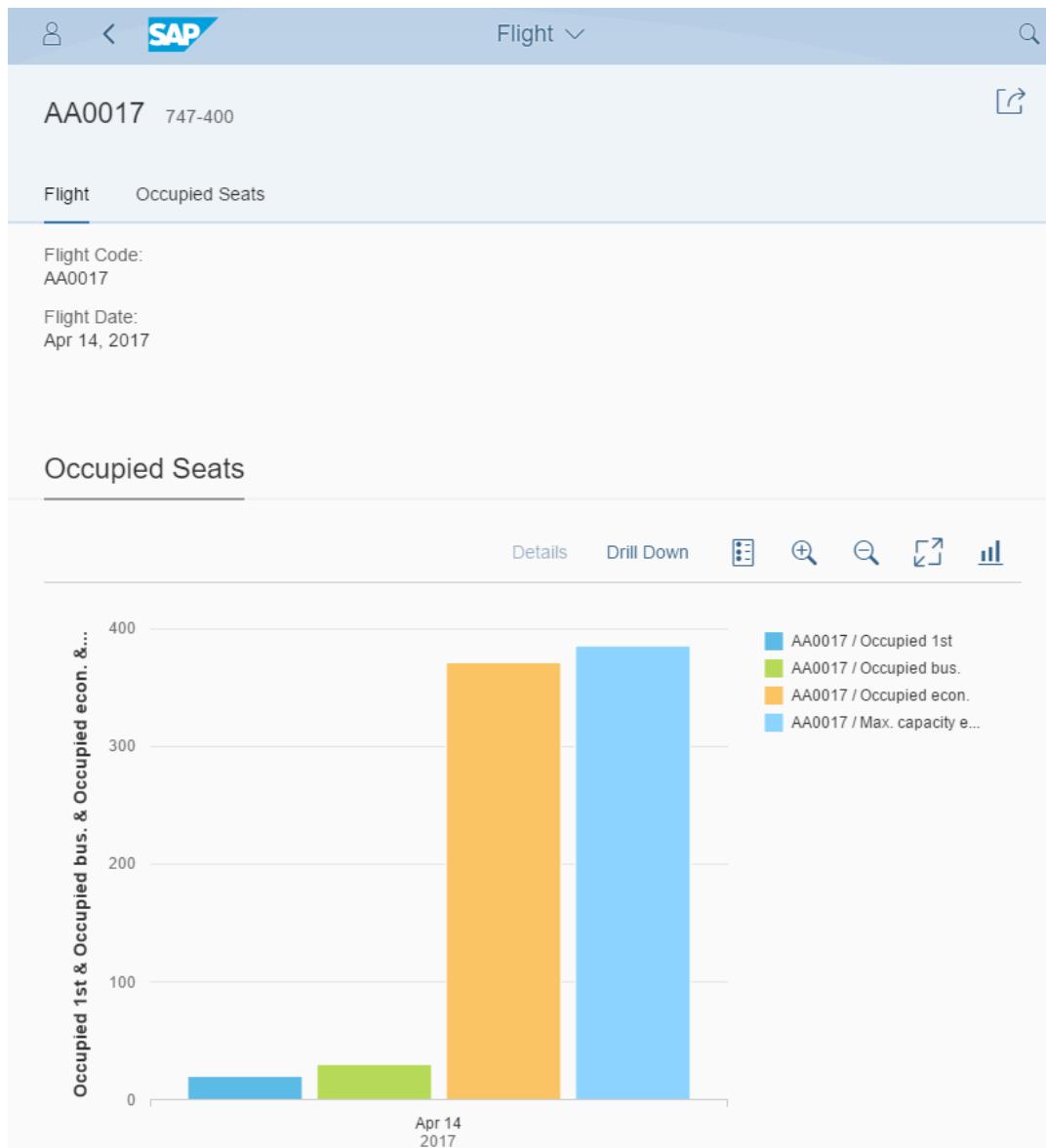
This is the expected outcome:

List Report:

The screenshot shows a SAP Fiori List Report application. At the top, there is a search bar labeled "Flight" and a filter bar with fields for "Flight Date" and "Flight Number". Below the header, there is a toolbar with icons for refresh, search, and export. The main area displays a table with the following data:

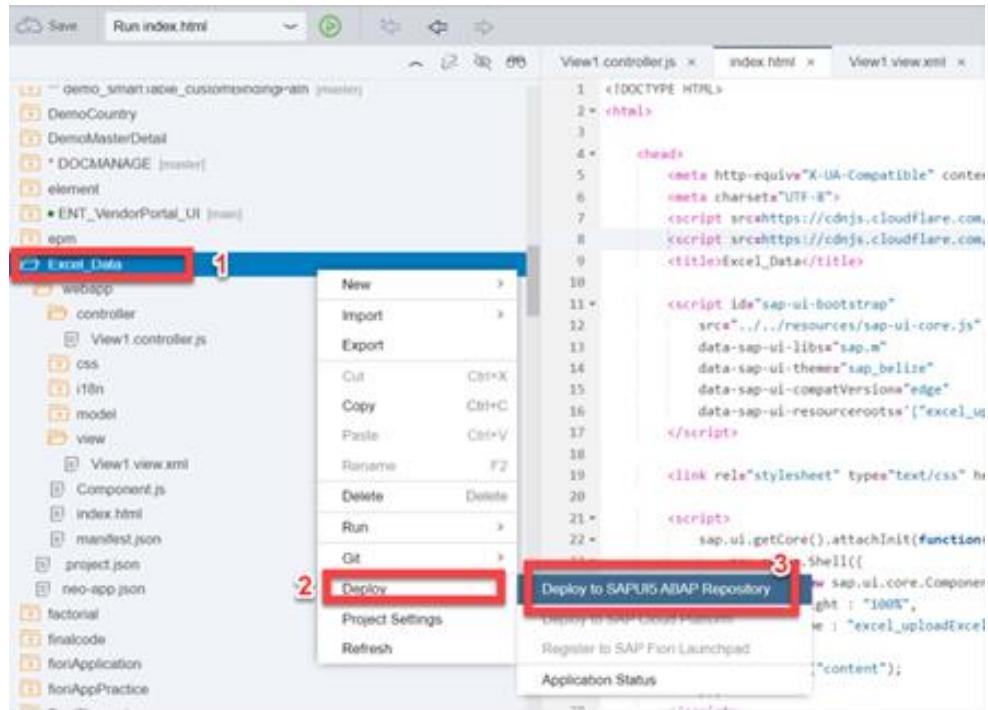
| Flight Code | Flight Date | Airline |
|-------------|--------------|------------|
| AA00011 | Aug 24, 2011 | 422504 USD |
| AA00011 | Aug 14, 2011 | 422504 USD |
| AA00011 | Jul 3, 2011 | 422504 USD |
| AA00011 | Sep 21, 2011 | 422504 USD |
| AA00011 | Dec 10, 2011 | 422504 USD |
| AA00011 | Feb 28, 2012 | 422504 USD |

Object Page:



Deploy the Fiori app using WEB-IDE:

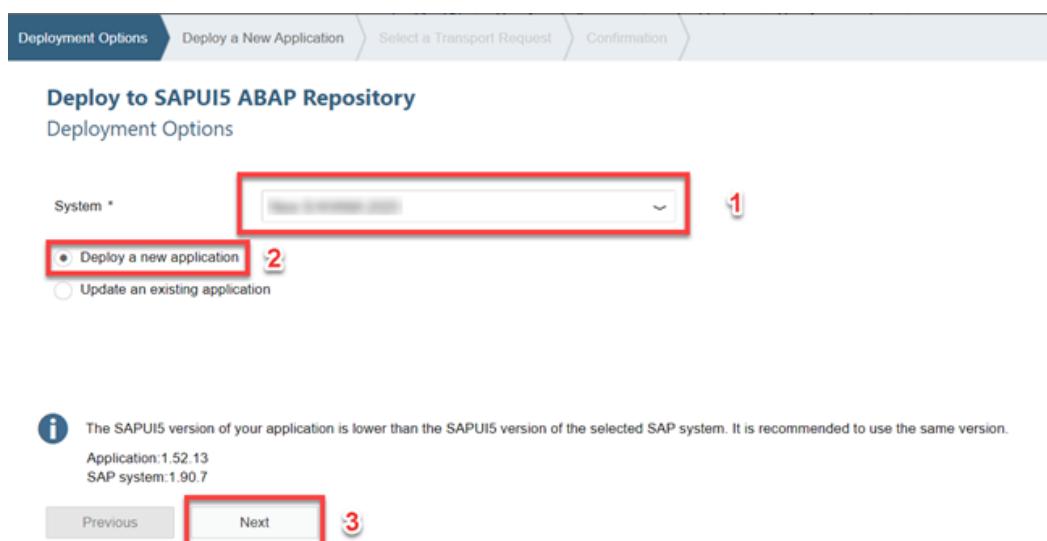
1. Go to the WEB-IDE, right-click on your project -> Select Deploy -> Choose your System (Deploy to SAPUi5 ABAP Repository)



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.

The screenshot shows the 'Deploy to SAPUI5 ABAP Repository' interface. At the top, there is a navigation bar with three tabs: 'Deployment Options', 'Deploy a New Application' (which is highlighted in blue), and 'Confirmation'. Below the navigation bar, the title 'Deploy to SAPUI5 ABAP Repository' is displayed, followed by the sub-section 'Deploy a New Application'. The form contains three input fields: 'Name *' with the value 'zexcel_upload', 'Description *' with the value 'Read data from excel file', and 'Package *' with the value '\$TMP'. A 'Browse' button is also present next to the package field. At the bottom of the form, there are 'Previous' and 'Next' buttons.

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

The screenshot shows the 'Deploy to SAPUI5 ABAP Repository' confirmation screen. At the top, there is a navigation bar with three tabs: 'Deployment Options', 'Deploy a New Application', and 'Confirmation' (which is highlighted in dark blue). Below the navigation bar, the title 'Deploy to SAPUI5 ABAP Repository' is displayed, followed by the sub-section 'Confirmation'. A message at the top states 'Click Finish to deploy your application to the SAPUI5 ABAP Repository.' At the bottom of the screen, there are 'Previous' and 'Finish' buttons, with the 'Finish' button being highlighted with a red rectangle.

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.

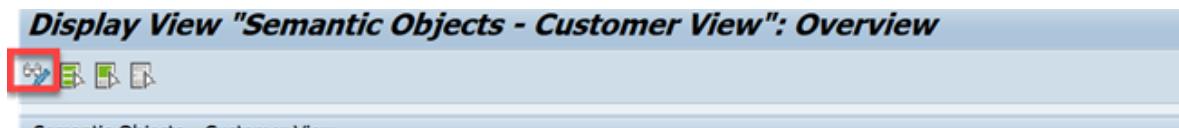
A screenshot of the SAP Repository Browser interface. The title bar shows "Repository Browser" and "MIME Repository". A dropdown menu is open, with "BSP Application" selected and highlighted with a red box. Below the menu is a toolbar with various icons. The main area is a table titled "Object Name" and "Description". One row is expanded, showing a folder named "ZEXCEL_UPLOAD" with a description "Read data from excel file". This row is also highlighted with a red box. The table has two columns: "Object Name" and "Description".

| Object Name | Description |
|-----------------|---------------------------|
| ▼ ZEXCEL_UPLOAD | Read data from excel file |

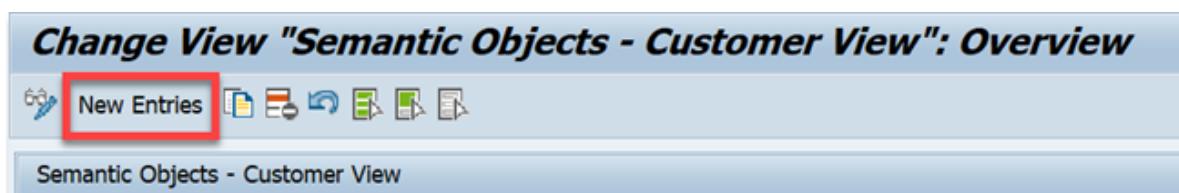
Steps to create a Tile and Target mapping in Fiori launchpad Designer

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

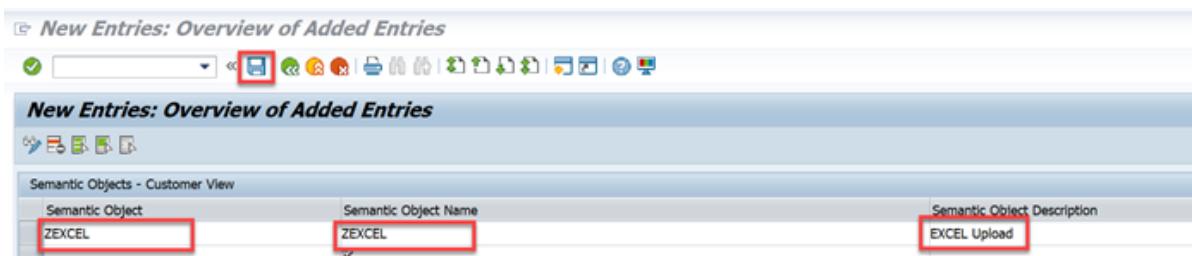
Click on Edit Button



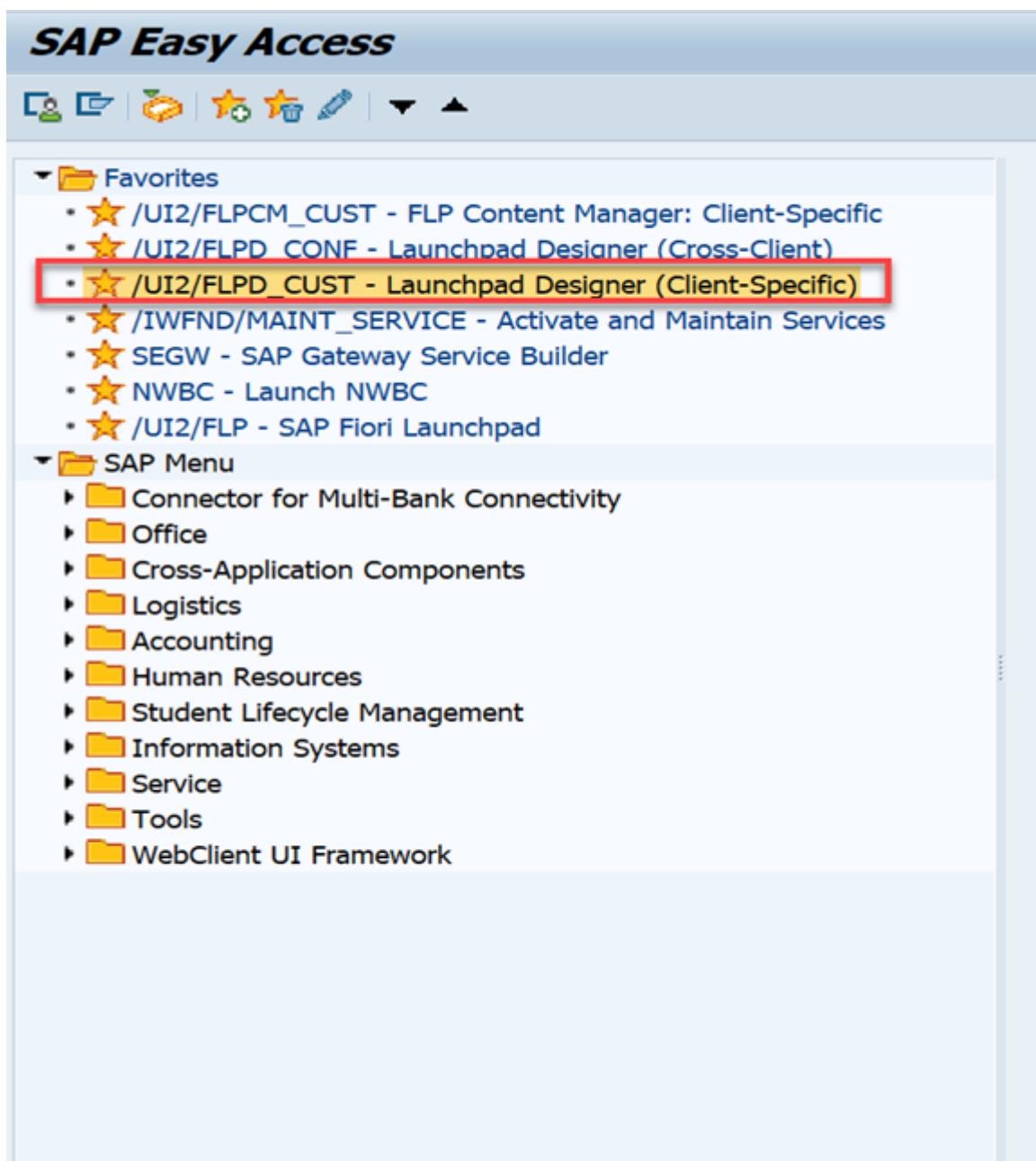
2. Select New Entrie.



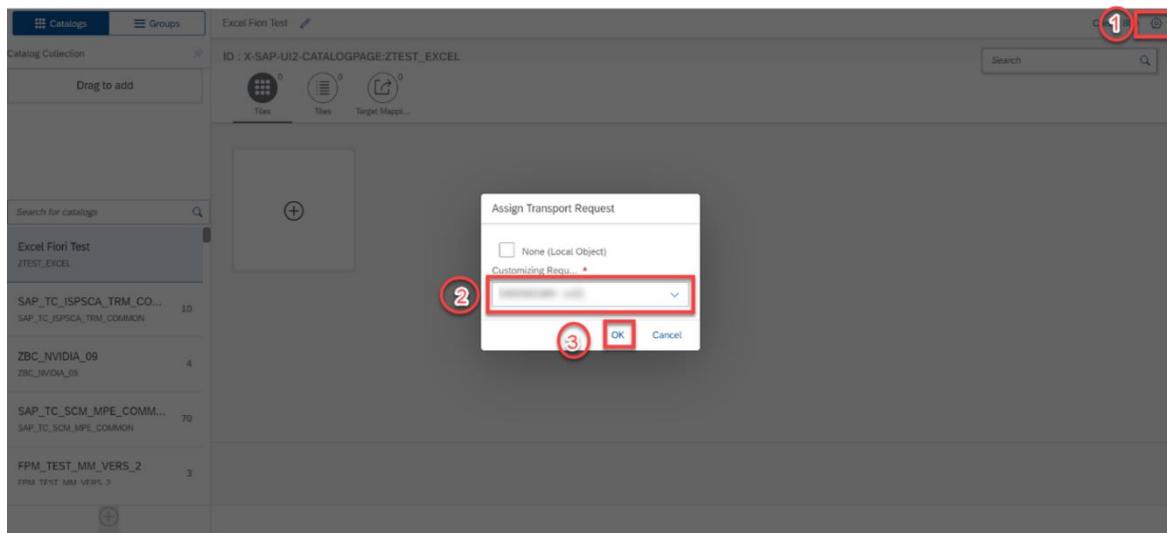
3. Provide the new Semantic Object, Object Name and Description and Click on Save Button.



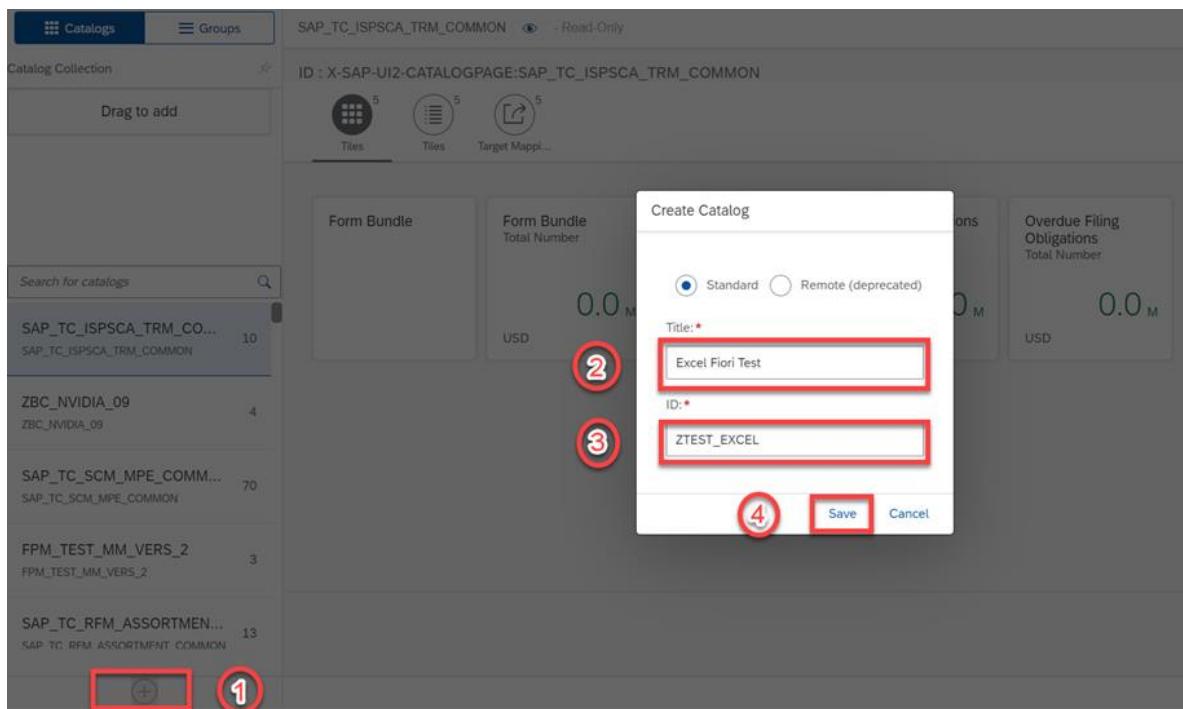
Go to SAP LogOn → Execute the TCode – **/UI2/FLPD_CUST** (Launchpad Designer).



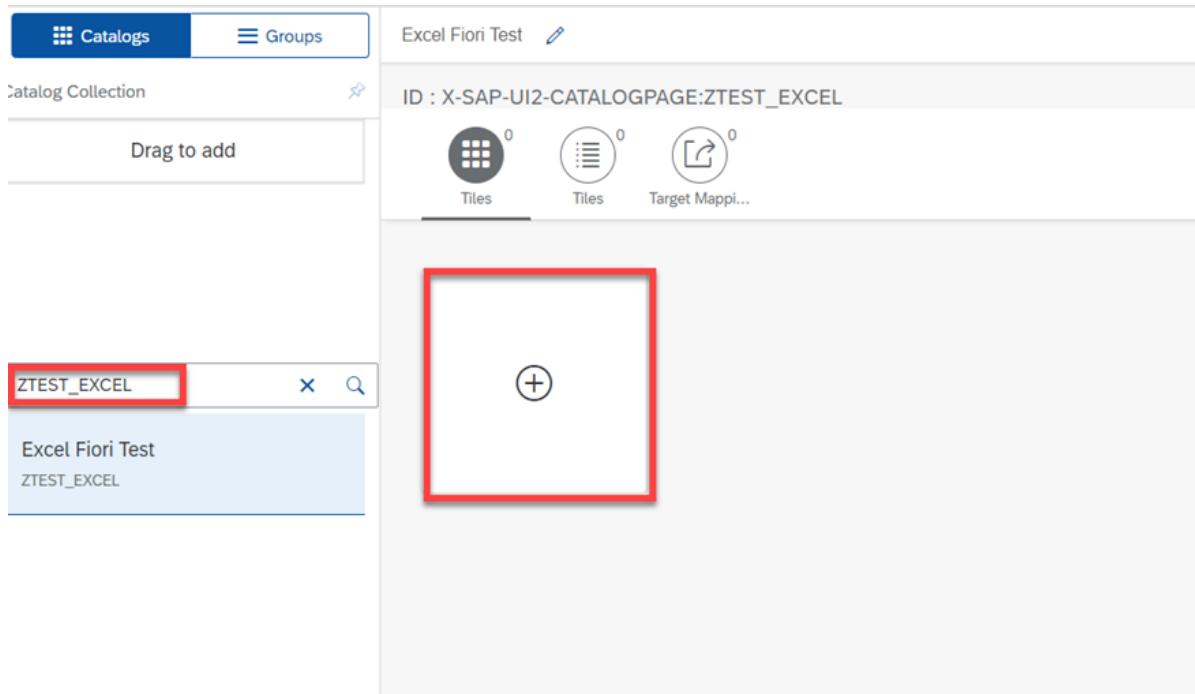
Click on Setting Button → Select the TR → choose Ok.



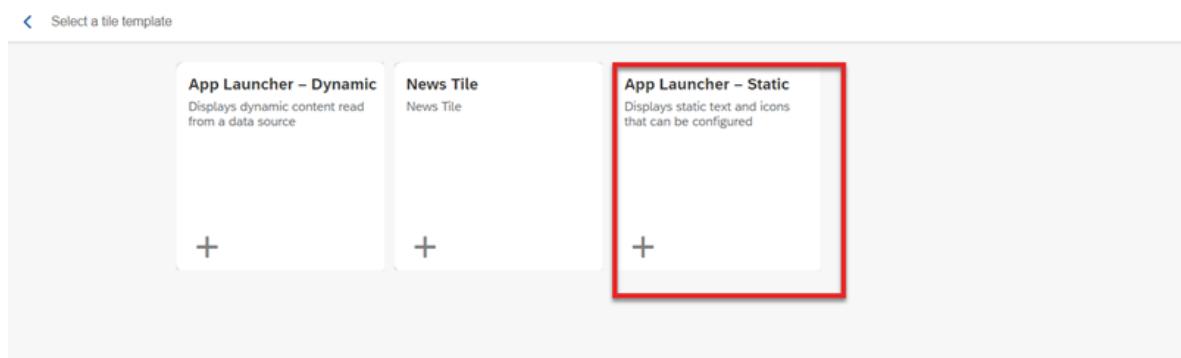
If catalog information is not available, create a catalog → Click on Plus Button → provide the required information → click on Save.



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.

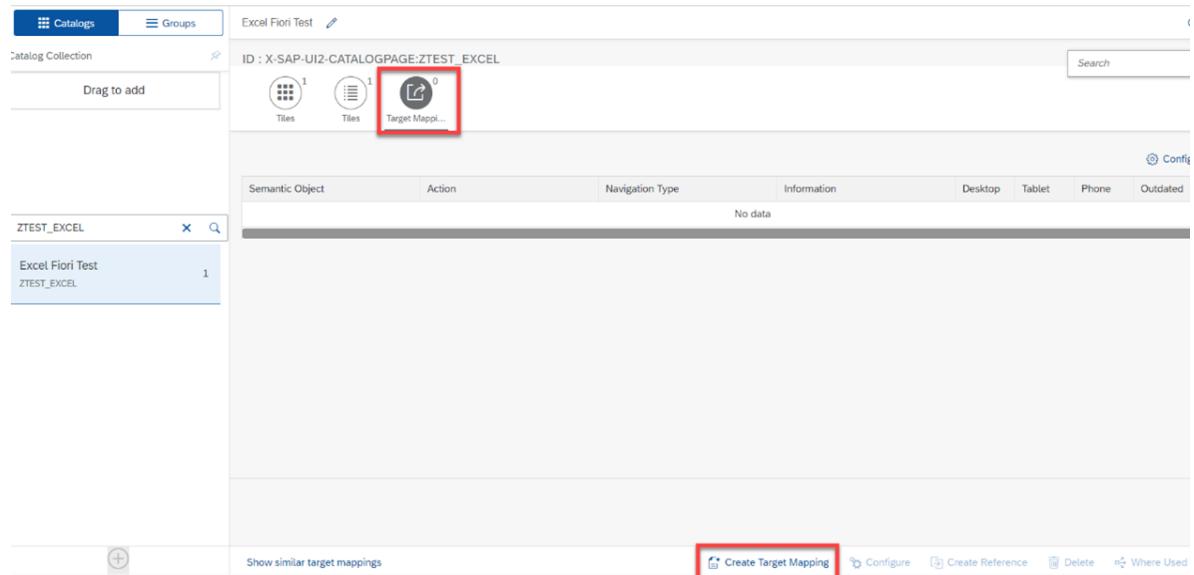


Provide the Title, information, Semantic Object, Action and other required Information Click on Save Button.

| | |
|------------------------|---|
| General | Navigation |
| Title: Excel Upload | Use semantic object navigation: <input checked="" type="checkbox"/> |
| Subtitle: | Semantic Object: ZEXCEL |
| Keywords: ZEXCEL | Action: change |
| Icon: sap-icon://inbox | Parameters: |
| Information: ZEXCEL | Target URL: |

| | | | | |
|------------------------------------|-------------|----------------------|---|------|
| Title Actions | | | | |
| <input type="checkbox"/> Menu Item | Target Type | Navigation Target | Action | Icon |
| <input type="checkbox"/> | URL | <input type="text"/> | <input type="button" value="sap-i..."/> | |
| | | | | |
| | | | | |

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.

The screenshot shows the SAP WEBIDE interface with multiple tabs open: View1.controller.js, index.html, manifest.json (which is the active tab), and View1.view.xml. On the left, there's a file tree showing a project structure with folders like 'dist', 'webapp', 'controller', 'css', 'i18n', 'model', and 'view'. Under 'manifest.json', the code is visible:

```

sap.uxap": {},
"contentDensities": {
  "compact": true,
  "cozy": true
},
"models": {
  "i18n": {
    "type": "sap.ui.model.resource.ResourceModel",
    "settings": {
      "bundleName": "excel_uploadExcel_Data.i18n.i18n"
    }
  }
},
"resources": {
  "css": [
    {
      "uri": "css/style.css"
    }
  ]
},
"sap.platform.abap": {
  "uri": "/sap/bc/ui5_ui5/sap/zexcel_upload/webapp",
  "version": "1.1.0"
}
}

```

A specific line of code, 'uri': '/sap/bc/ui5_ui5/sap/zexcel_upload/webapp', is highlighted with a red box.

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

The screenshot shows two parts of the Fiori app configuration:

- manifest.json:** A code editor view showing the JSON configuration for the app. The line `id: "excel_uploadExcel_Data",` is highlighted with a red box.
- Configure: 'Target Mapping':** A dialog box for defining the target for the app. It includes fields for Intent (Semantic Object: ZEXCEL, Action: change) and Target (Application Type: SAPUI5 Fiori App, Title: Excel Upload, URL: /sap/bc/ui5_ui5/sap/zexcel_upload, ID: excel_uploadExcel_Data).

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 → we must assign the PFCG Role for catalog and group and assign that role to User.

The screenshot shows the SAP Fiori Launchpad interface with the following details:

- Header:** SAP App Finder
- Search Bar:** All | excel
- Catalog:** Catalog | User Menu | SAP Menu
- Search Result:** Excel
- App List:** A list of apps including "Excel Fiori Test" (highlighted with a red box), "Engineering Record", "Enterprise BOM", etc.

Conclusion:

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

