Inter Application Communication and Navigation (Design Time) in WebDynpro

Applies to:

SAP NetWeaver 7.0 (EP7). For more information, visit the Web Dynpro Java homepage

Summary

As we all know how to create a WebDynpro application to execute various requirements using multiple views. We just need to create a Development Component and create components and views as required. But if the requirement is huge and it needs to be delivered within a very short time, then multiple developers are required to work in order to finish the work on time. But the problem is – only one developer can work in one workspace at a time. The project needs to be split across different modules (i.e. different development components) that will allow multiple developers to work independently. As the integrated project has been split into multiple small components, these components need to communicate with each other in order to achieve the goal. Inter component communication and navigation can be done in two ways – i) Run time ii) Design time. This article provides step by step guide to integrate between multiple WebDynpro components (design time) such that communication and navigation between them becomes possible.

Author: Mr. Sudip Majumder

Company: TCS – Tata Consultancy Services

Created on: 18 August 2009

Author Bio

Sudip Majumder is SAP certified - Development Consultant (SAP NetWeaver 2004 - Portal &KMC). He is working in TCS since 2004.

Table of Contents

Problem Description

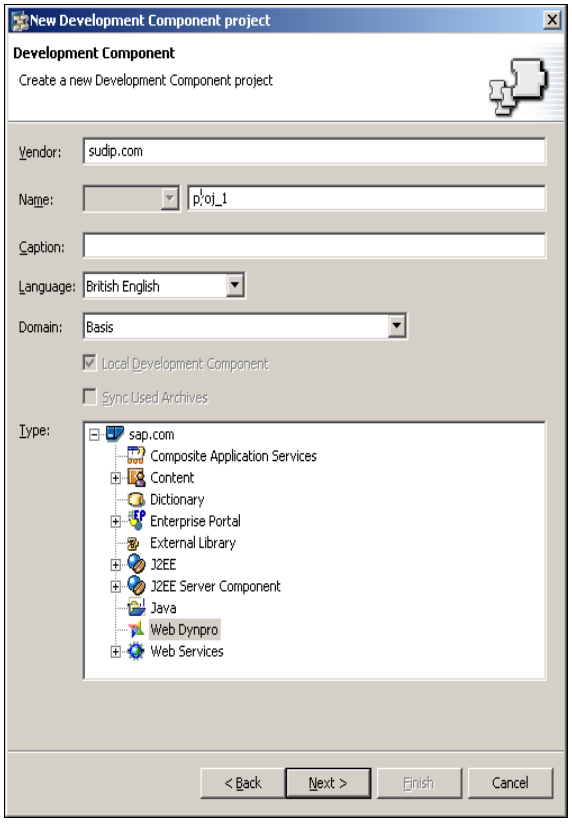
As we all know how to create a WebDynpro application to execute various requirements using multiple views. We just need to create a Development Component and create components and views as required. But if the requirement is huge and it needs to be delivered within a very short time, then multiple developers are required to work in order to finish the work on time. But the problem is – only one developer can work in one workspace at a time. The project needs to be split across different modules (i.e. different development components) that will allow multiple developers to work independently. As the integrated project has been split into multiple small components, these components need to communicate with each other in order to achieve the goal. Inter component communication and navigation can be done in two ways – i) Run time ii) Design time. This article provides step by step guide to integrate between multiple WebDynpro components (design time) such that communication and navigation between them becomes possible.

Solution Details

Following steps need to be performed in order to communicate and navigate between WebDynpro applications -

Creating Development Component

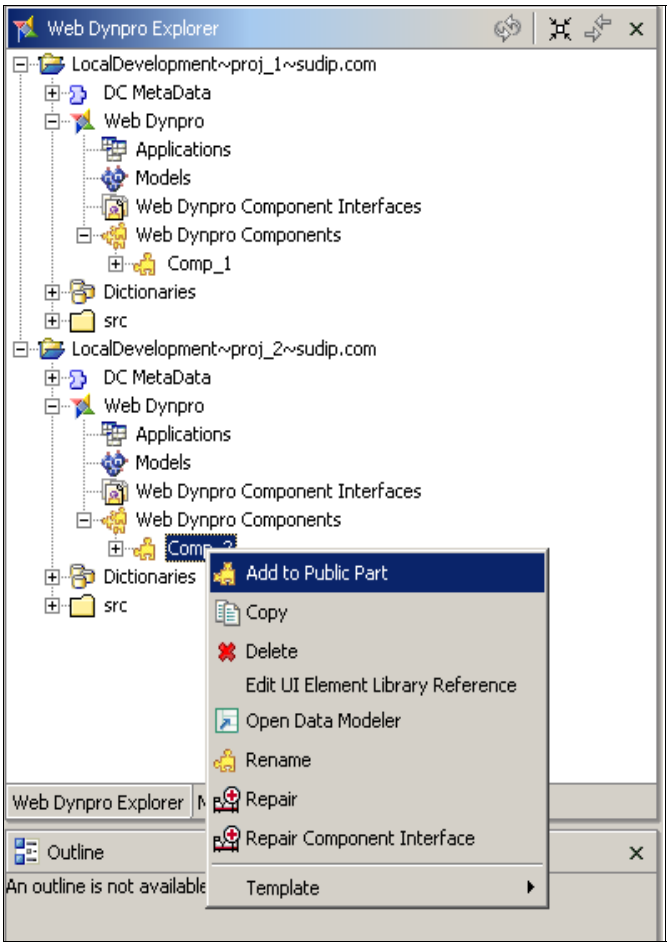
- Create a Development Component project (of type WebDynpro). Name it as ***proj_1***. This will act as the start component, i.e. it will send some data to another application. Create another Development Component project (of type WebDynpro). Name it as ***proj_2***. This will act as the target application, i.e. it will receive data sent from the source application.



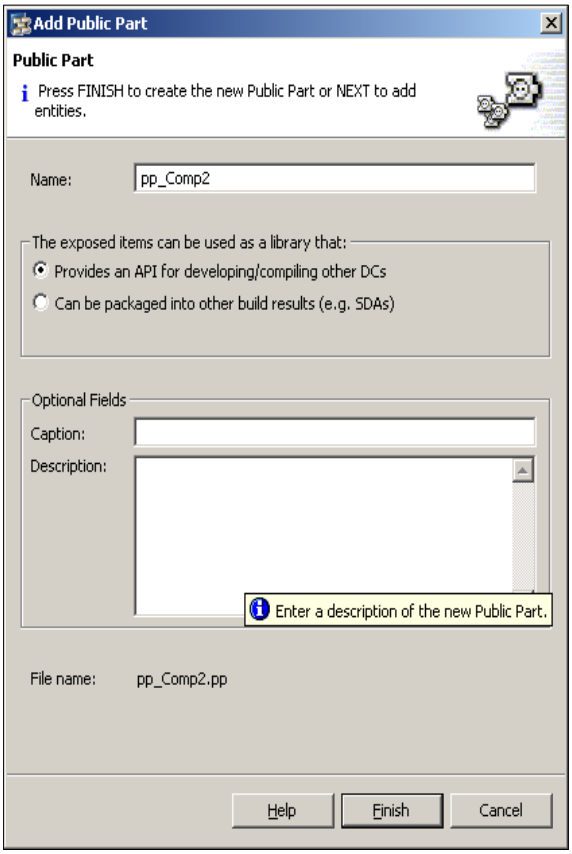
- Create WebDynpro components ***Comp_1*** and ***Comp_2*** respectively in proj_1 and proj_2.

Adding to Public Part

- Define the component (Comp_2) for public usage. This will allow other components to access this component. In the WebDynpro Explorer, right click on the Comp_2 component and select "*Add to Public Part*".

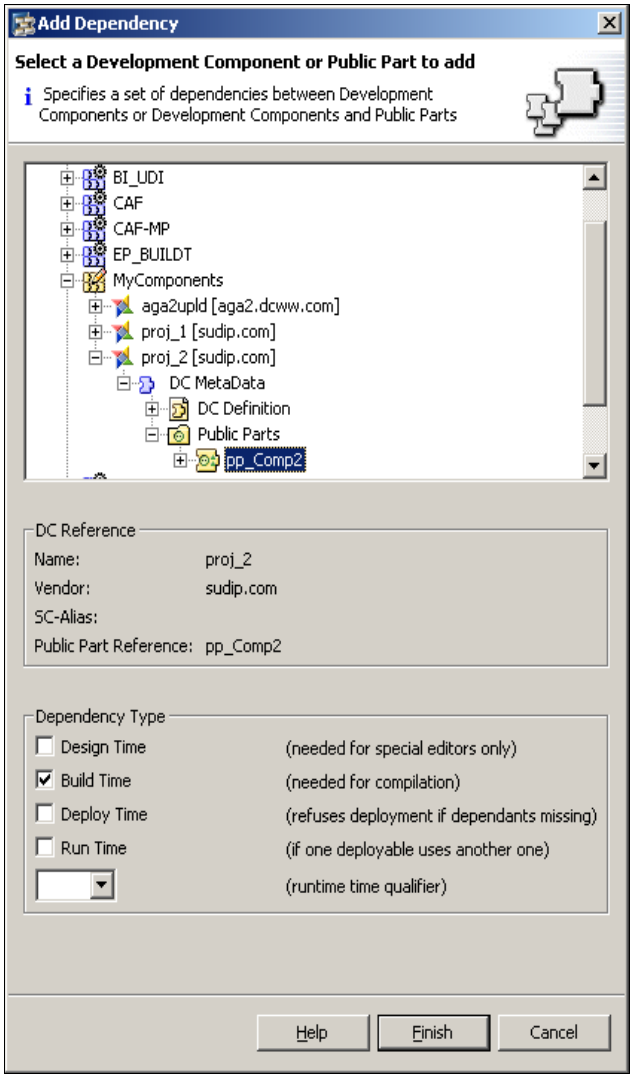


- Provide a suitable name for the public part (e.g. pp_Comp2). Select for developing/compiling other DCs.



Build the Development Component. Right click on the project → Development Component → Build.

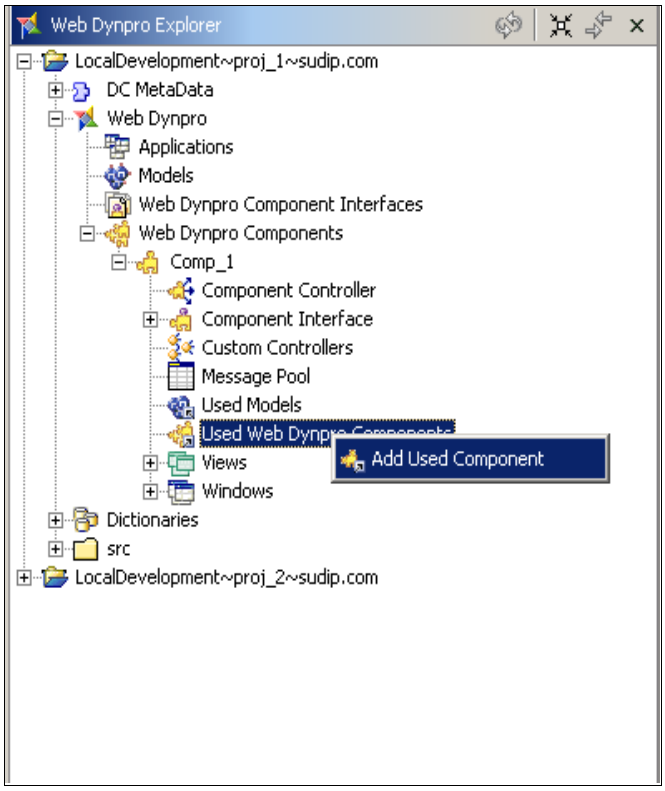
- Comp_2 is now ready for re-use from another Development Component. Open the component; navigate to DC Definition → Used DCs. Right click on Used DCs and click Add Used DCs. In the next window, select the public component in Comp_2 (pp_Comp2) and click Finish.



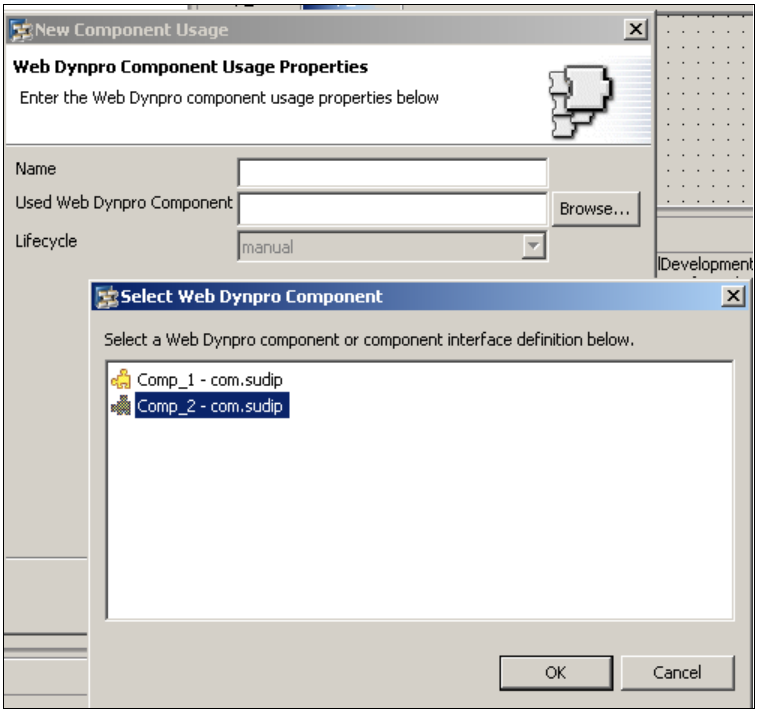
- Build the Development component of proj_1 (Right click on proj_1 → Development Component → Build).

Adding Used WebDynpro Component

- Open the navigation tree (in WebDynpro explorer) and navigate to proj_1. Navigate to Web Dynpro → Web Dynpro Components → Comp_1 → Used Web Dynpro Components. Right Click and select "***Add Used Component***"



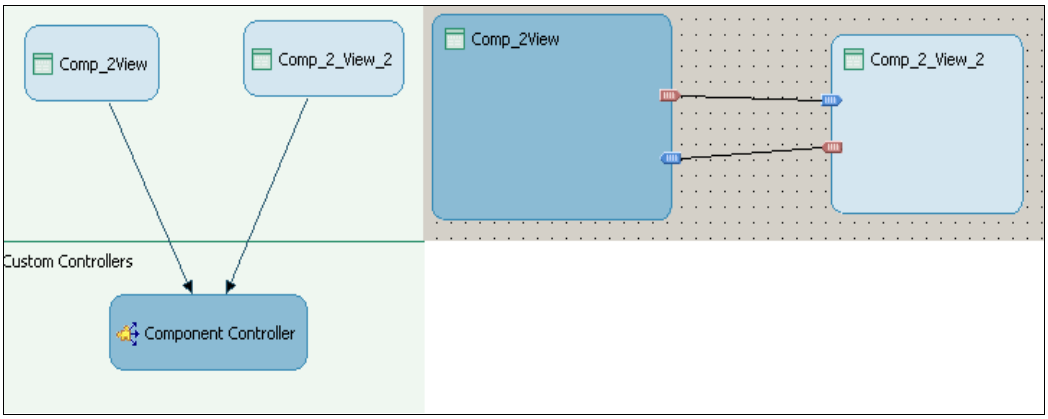
- Click Browse and select Comp_2



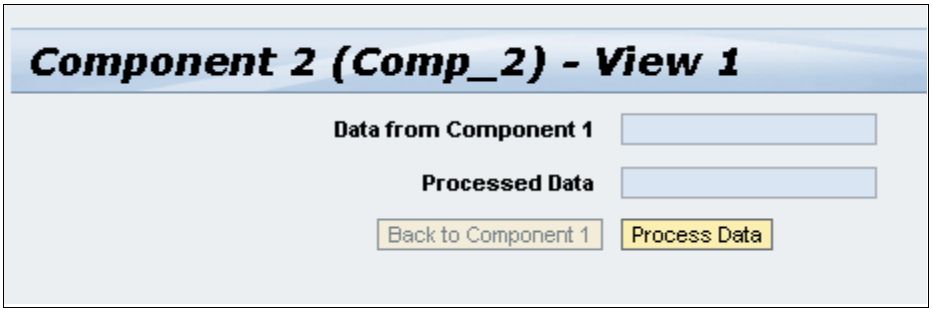
- Provide name as Used_Comp_2 and select "*createOnDemand*" option. Click on Finish. Interface to Comp_2 is now available to Comp_1 and it can be re-used now.

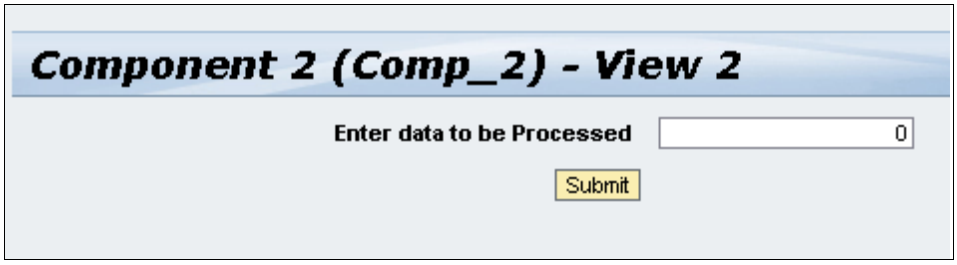
Developing Components Functionality

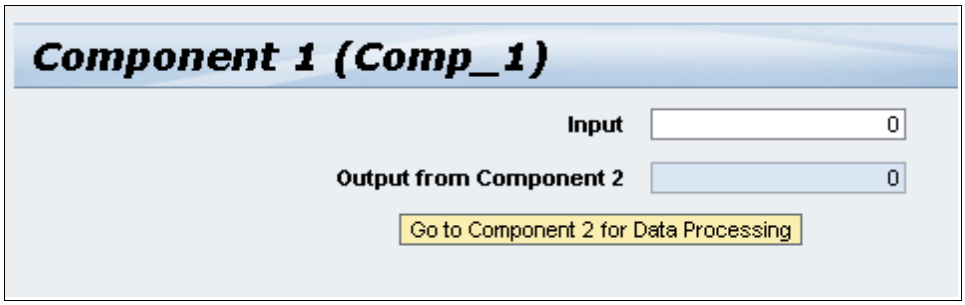
- Modify the components (Comp_1 and Comp_2) to achieve the required functionality of each of the components individually.
- Add views, actions etc. as per the desired requirements. Define Inbound and outbound plugs (if required) within the component. Use component controller for data communication within the views (if required). Comp_2 has been created as per the diagram mentioned below -



- Comp2 has been developed to process some data. It expects one data as input (from another component) and one input has to be given explicitly at Comp2. Image is given below -



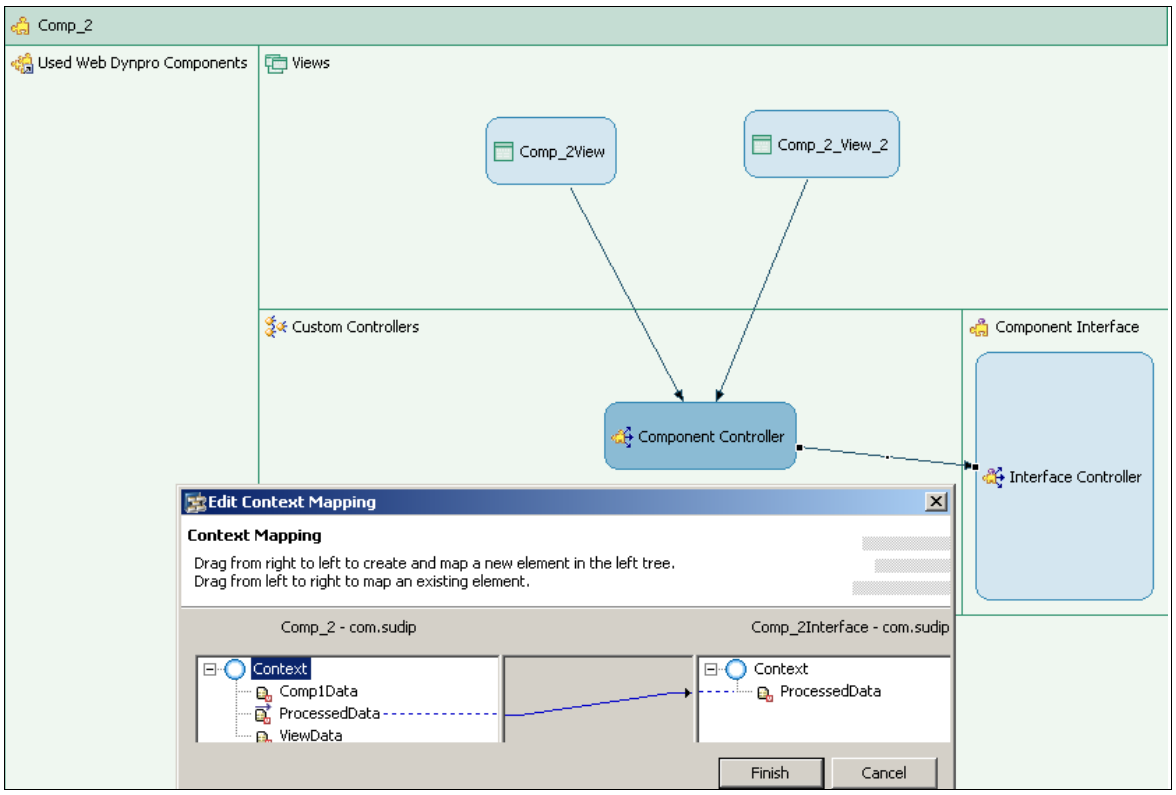




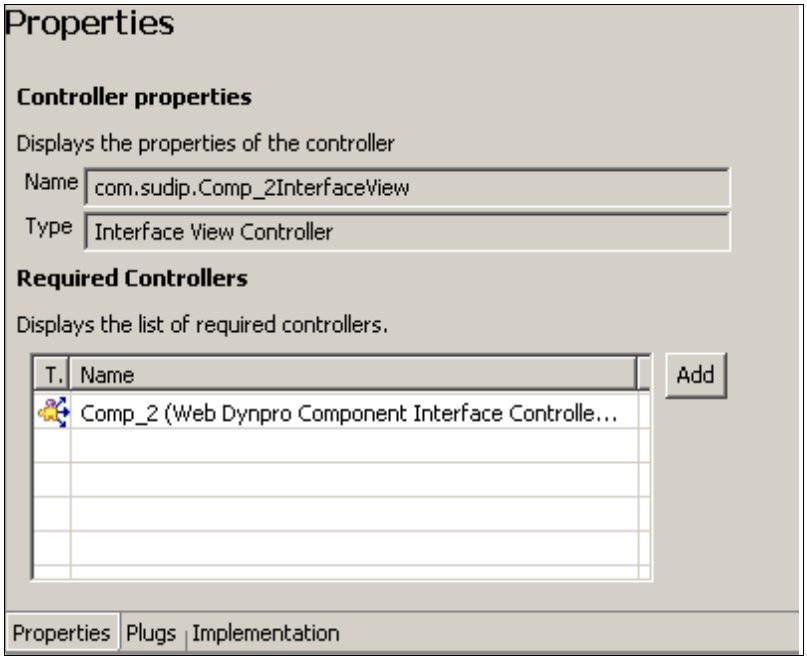
- Comp_1 is a simple component. It provides one data as input to Comp_2 and gets output from it and proceeds further.

Enabling Inter Component Navigation and Communication

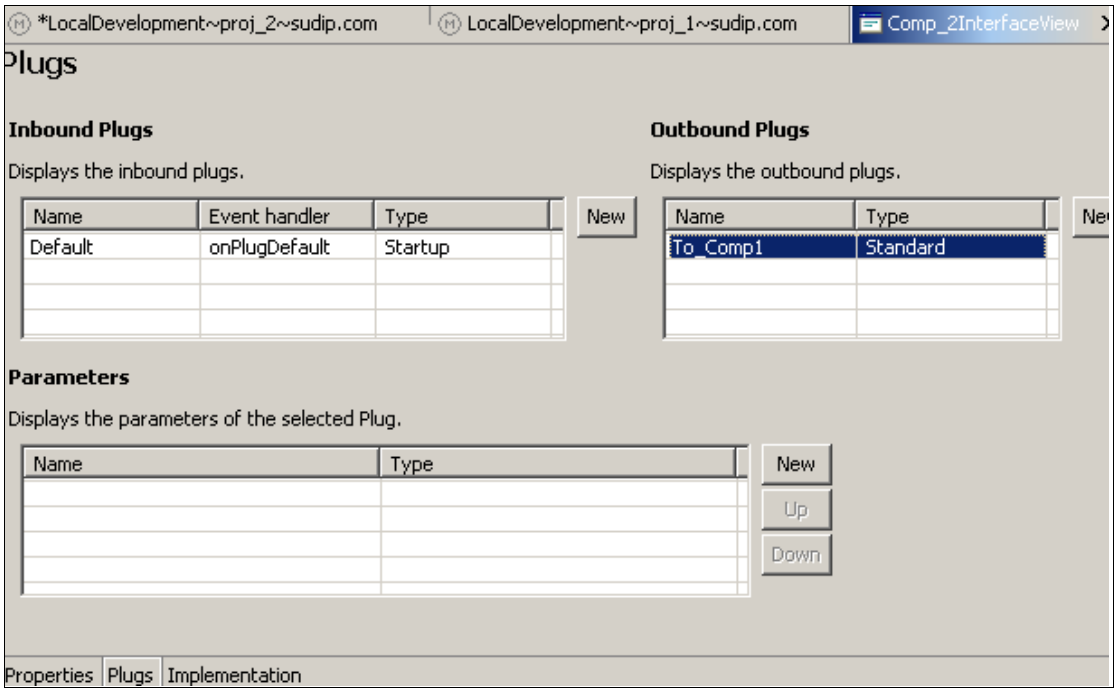
- As both components (Comp_1 and Comp_2) need to interact with each other to achieve the desired functionality, these components needs to be integrated.
- Define Interface Controller for Comp_2 and make connection with the component controller of Comp_2. ProcessedData attribute has been bound to the component controller. The image below will describe the same -



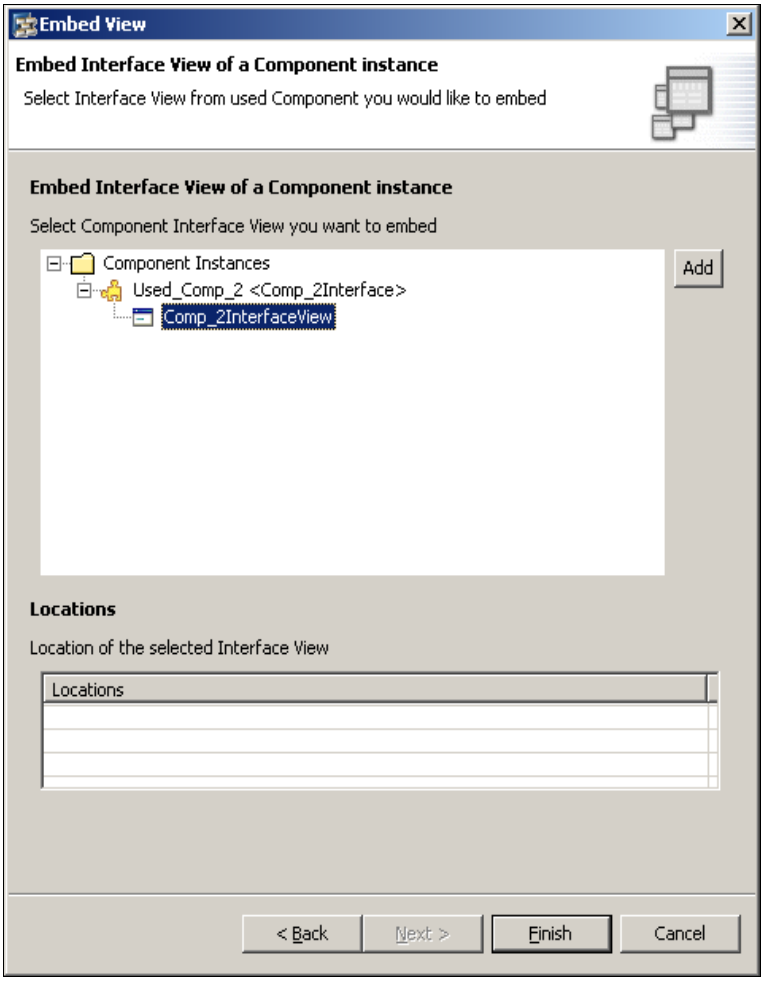
- Navigate to proj_2 → Comp_2 → Component Interface → Interface Views → Comp_2InterfaceView and open the same. In the *Properties* tab, add Comp_2 Interface controller.



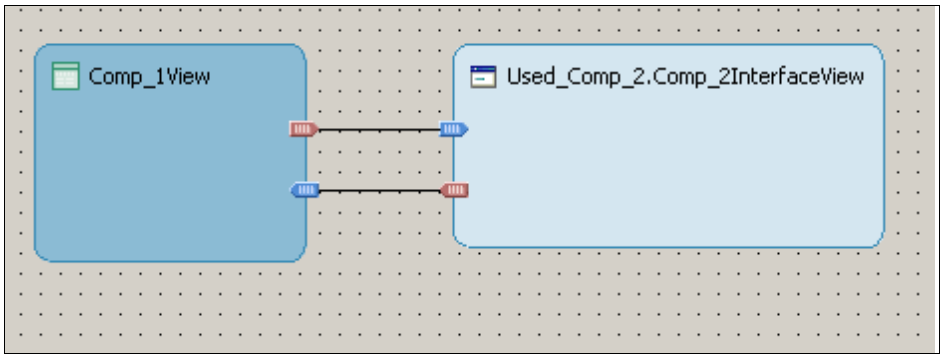
- Go to *Plugs* tab. Add an standard Outbound Plug (by clicking *New* button). Name it as To_Comp1 (shown in the image below).



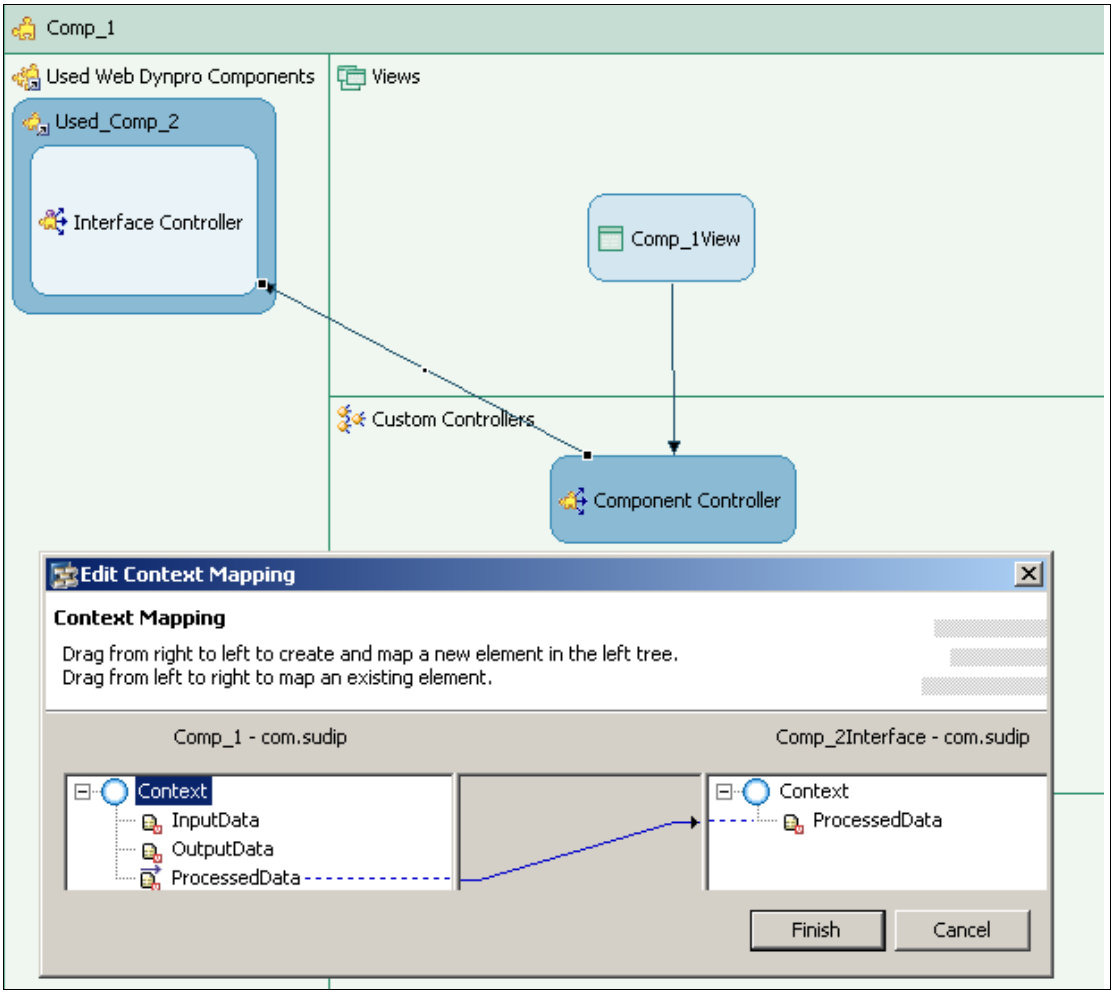
- Build the Comp_2 Development Component.
- Open Comp_1 and open Comp_1 Windows.
- Embed a view in the window. Select "*Embed Interface View of a Component instance*" in the Embed View menu.
- Select Comp_2 Interface View in the next screen and click *Finish*.



- Interface View is now placed in the Window. Inbound (available by default) and outbound (To_Comp1) plugs will be present in the interface view.
- Create Inbound and Outbound plugs in Comp1View and create navigation links between Comp1View and the interface view. Image given below -



- Define connection with the component controller of Comp_1 and Used Comp_2 Interface Controller. ProcessedData attribute has been bound to the component controller. The image below will describe the same –



- Similarly bind InputData with the Interface View of the Comp_1 and Comp_2 (as described for ProcessedData). The data flow will be –

  Comp_1View → Comp_1 controller → used Comp2 interface controller → Comp_2 Interface Controller → Comp_2 Controller → Comp_2 View 1.

- After reaching component 2, the data flow will be -

  Comp_2 View1 → Comp_2 controller → Comp2 View 2 → Comp_2 controller → Comp2 View 1 → Comp_2 Interface Controller → used Comp2 interface controller → Comp_1 View.

- Define the navigation by creating actions and the fire plugs as created in the steps mentioned above.

- Create a method in Comp_2 controller. Name it as *BackToComp1.* Add Interface View Controller as Required Controller in Comp_2 (in Properties tab). Implement the function as -

```
public void BackToComp1( )
{
  //@@begin BackToComp1()
  wdThis.wdGetComp_2InterfaceViewController().wdFirePlugTo_Comp1();
  //@@end
}
```

- Create an action in Comp_2 View 1, name it as To_Comp_1. Assign this action to the button "*Back to Component 1*". Implement the function as -

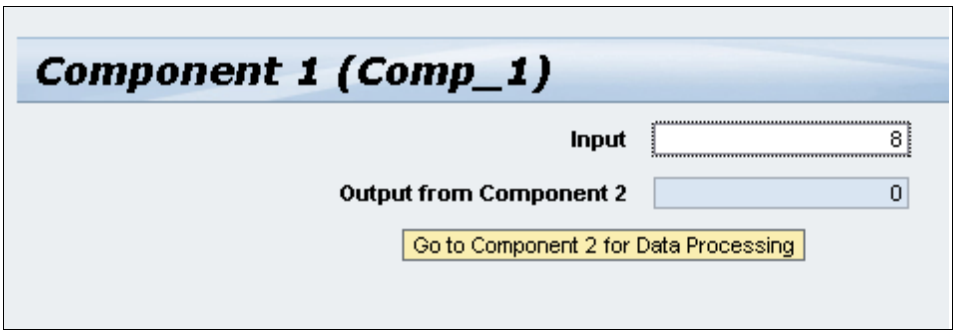
```
public void onActionTo_Comp_1(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent )
  {
    //@@begin onActionTo_Comp_1(ServerEvent)
    wdThis.wdGetComp_2Controller().BackToComp1();
    //@@end
  }
```

- Build Comp_2 and Deploy. Create an application for proj_1 (as it will be the starting point of the application). Build Comp_1 and Deploy.

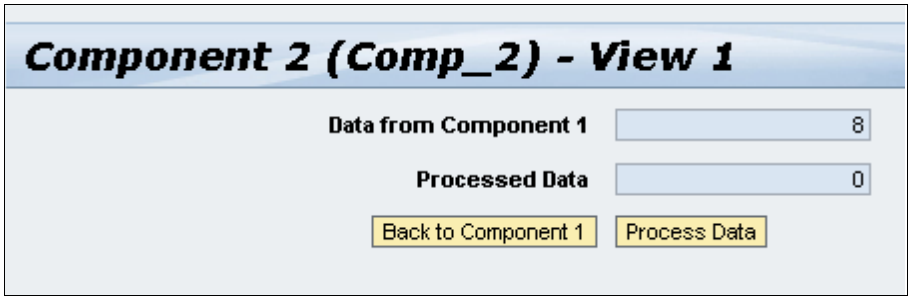
Application Flow (Data and Navigation)

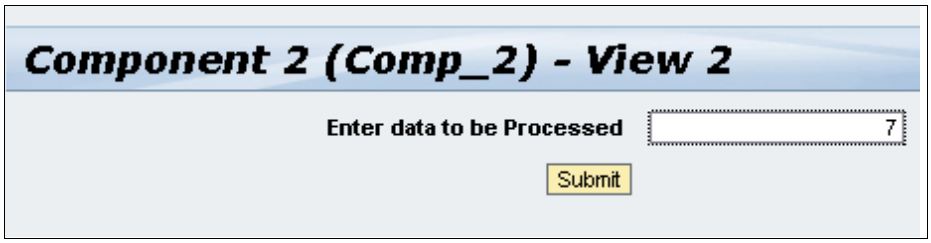
The flow of the above created application will be -

- First Screen (Comp_1). Provide Input as "8". Click on "*Go to Component 2 for Data Processing*"

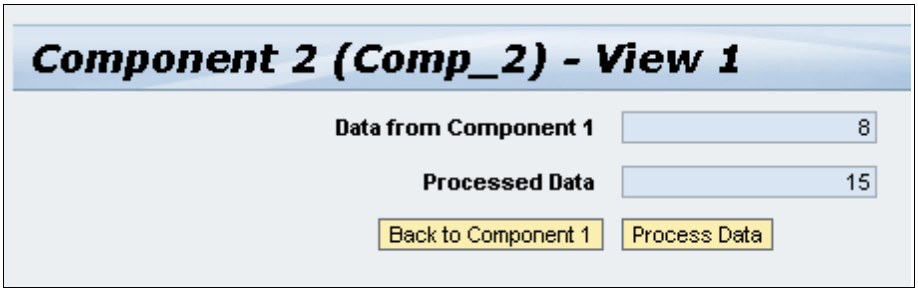


- View 1 of Comp_2 is shown. Data from Comp_1 (8) has been transferred to Comp_2 for processing. Click on "*Process Data*".

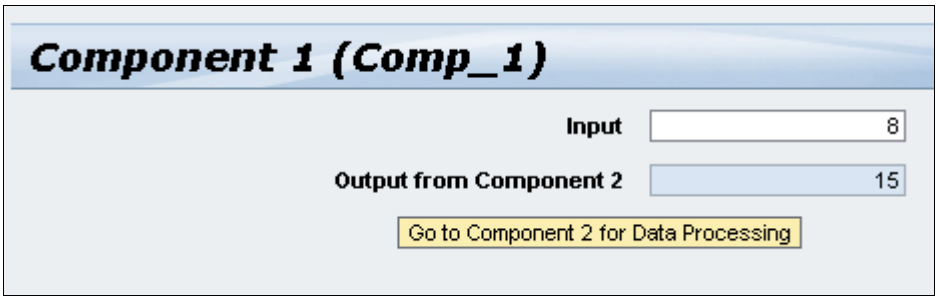




- View 2 of Comp_2 is shown. Enter "7" as the next data for processing. Click on "*Submit*"
- Data is processed and updated data is shown in Comp_2 View 1. Click on "*Back to Component 1*".



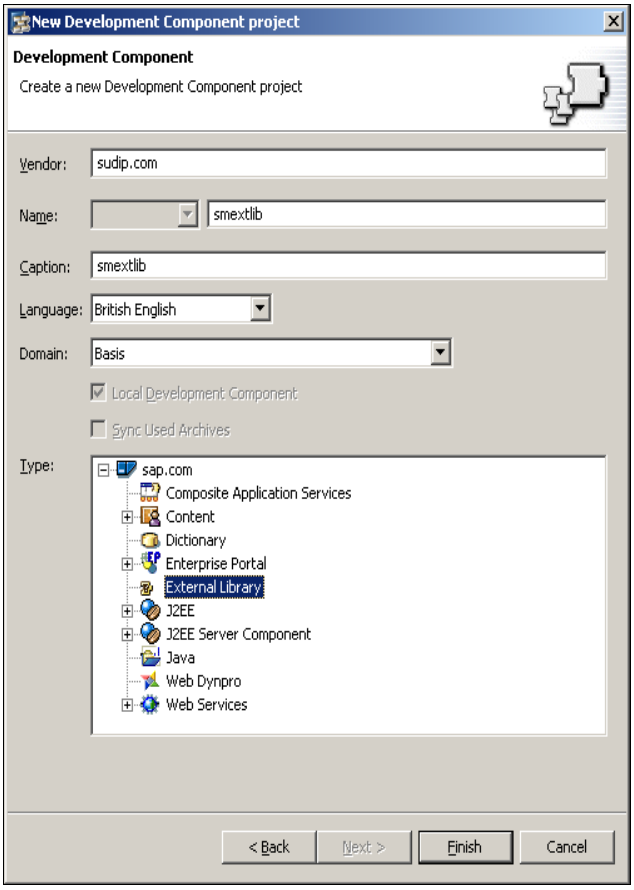
- Comp_1 view is displayed. Data from Comp_2 (15 i.e. Processed Data) has been communicated to Comp_1.



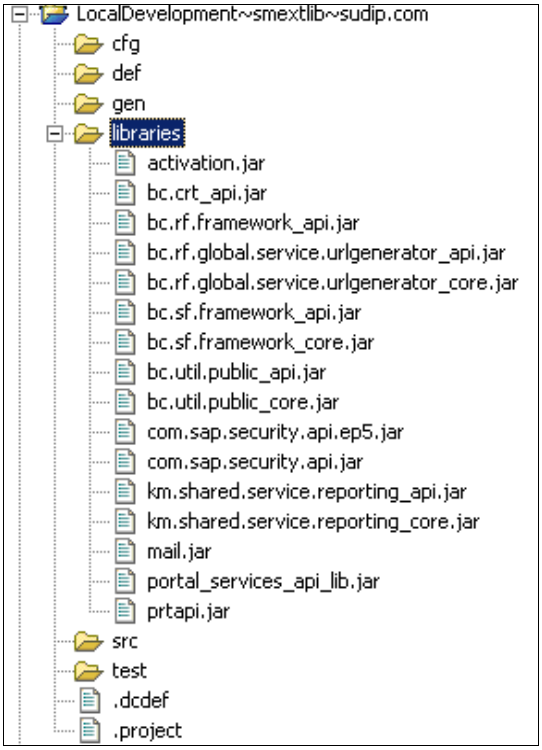
Using External Libraries

It is not recommended to use jar files inside the workspace in the Development Components. It is suggested to use an external library and add all the required jar files for the whole project. This external library can be made as a public component and all the Development Components can use the same. Follow the below mentioned steps to achieve the same -

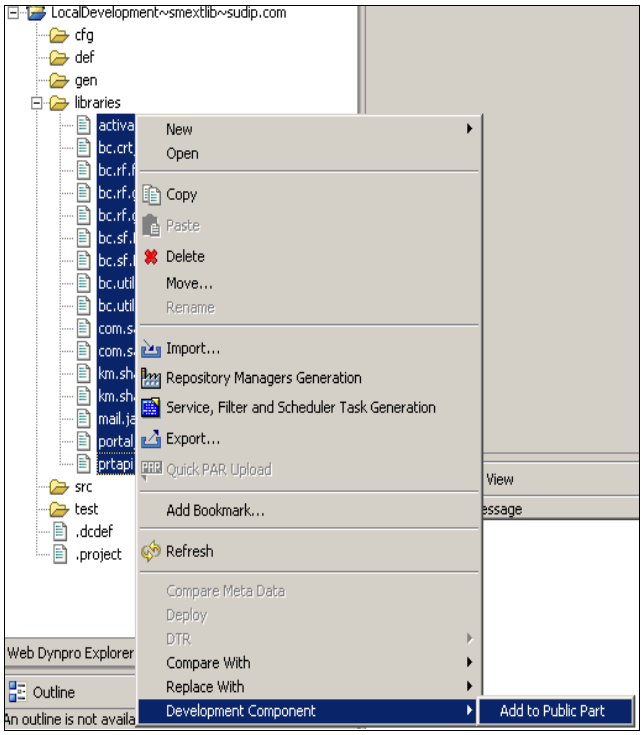
- Click File → New → Development Component Project. Provide suitable name (*smextlib*) and namespace (*sudip.com*) and select Type as "*External Library*" as shown in the image below -



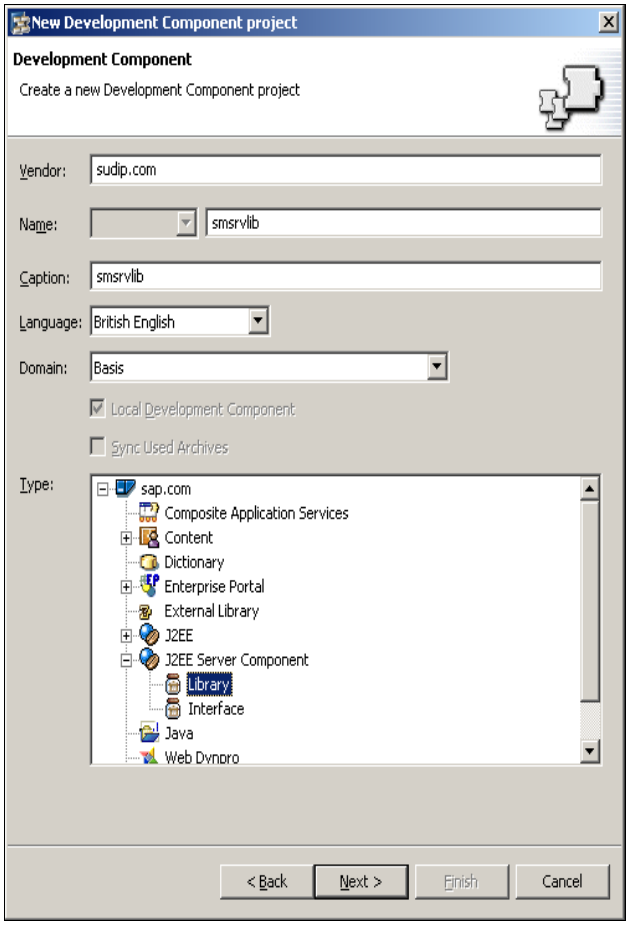
- Click Finish.
- Go to Navigator explorer and paste all the related jar files in *libraries* folder (shown in the image below) -



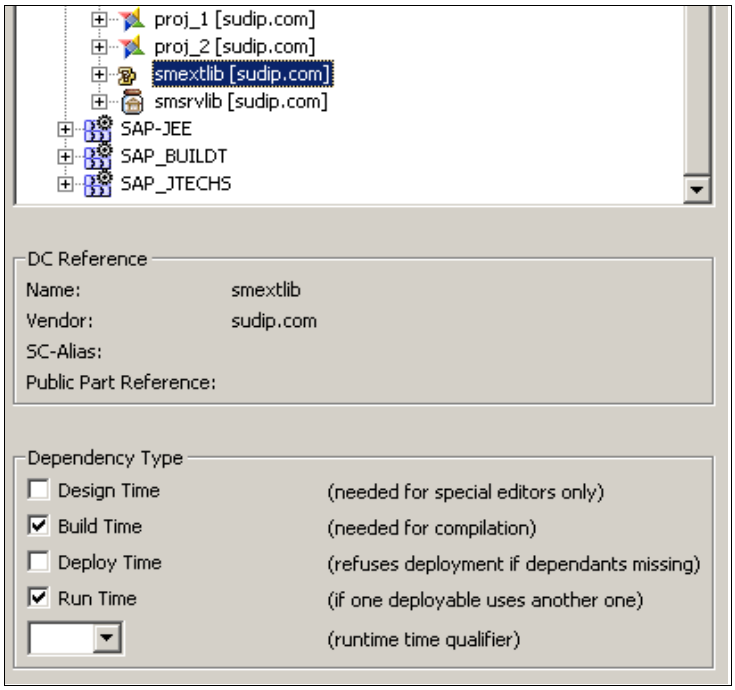
- Select all the jar files and add them to Public Part (Right click → Development Component → Add to Public Part)



- Provide name as pp_lib and select "*Provides an API for developing/compiling other DCs*". Re-do the above mentioned step again, name it as pp_lib_assy and select "*can be packaged into other build results (e.g. SDAs)*".
- To use this library, this public components need to be added as Used DCs in development components (DC Metadata → DC Definition → Used DCs → Add Used Dcs).
- To use this library centrally, this library needs to be deployed in the server as well. So creating a server component for this library is necessary. Click File → New → Development Component Project. Provide suitable name (*smsrvlib*) and namespace (*sudip.com*) and select Type as "*J2EE Server Component → Library*" as shown in the image below -



- Click Next and Finish.
- Open the project *smsrvlib*. Open DC Metadata and Right click on Used DCs and click Add Used DCs. Browse and select smextlib and check Design and Run time (as shown in the image below).



- Click on Finish.
- Right click on the Server component → Development Component → Build.
- After successful build, Right click on the Server component → Development Component → Deploy.
- Now all the components should run fine using the external library. To manually deploy them in other environments, get the ear files (for the DCs) and sda file (for the library) in the *deploy* folder in your workspace.

Related Content

sdn.sap.com

help.sap.com

Inter Application Communication and Navigation in Web Dynpro (Run time)

For more information, visit the Web Dynpro Java homepage

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document..