Generic User Interface Services Exercise

Chapter: **Generic UI Services**

Theme: **Extended Value Selector**

At the end of this Exercise, you are able to:

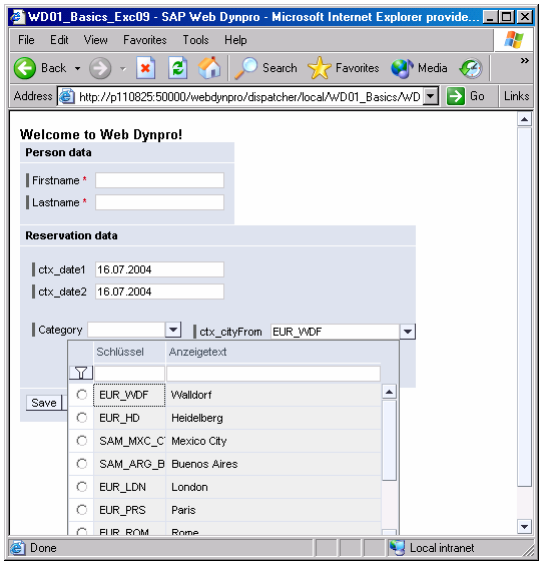
- Define Simple Data Types in the Java Dictionary.
- Create an Extended Value Selector.
- Change the values of the Value Selector programmatically.

1 Development Objectives

Value Help

The Web Dynpro runtime environment provides generic UI services for the application development. They enable you to create a value help for constants easily – for example, for countries or zip codes. The extended value selector can be used as a value selector for a large number of constants. Inserting an extended value selector is similar to inserting a simple value selector. The difference is that instead of a DropDownByKey UI element, an InputField UI element is bound to a value attribute whose data type is of the type Simple Type. You can define the data type Simple Type in the Java Dictionary at design time or you can create it using the ISimpleTypeModifiable interface for modifying the data type of a value attribute at runtime. This interface does not provide the required metadata until runtime. The main binding mechanisms are retained in this process.

2 Result



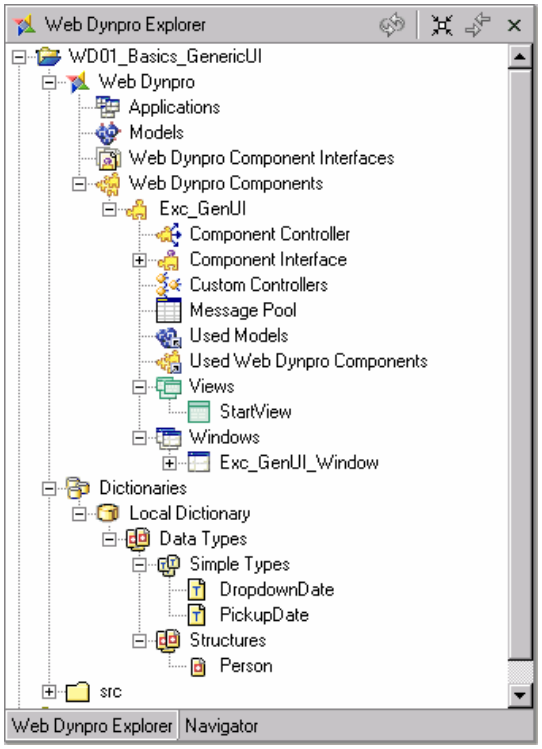
In this exercise, you will develop an extended value selector and fill it with data (City names for pickup and drop down locations) that you've defined in the Java Dictionary at design time.

Optional part:

You will learn how value attributes can be filled dynamically with dynamic metadata.

3 Prerequisites

You have launched the SAP NetWeaver Developer Studio.
You have selected the Web Dynpro perspective.
You have opened the *WD01_Basics_GenericUI* project.
The structure this project is currently displayed in the *Web Dynpro Explorer*.



For your convenience, you can start developing with a predefined view.

Expand the *Exc_GenUI* node. The graphic on the left shows the predefined project structure of this exercise.

4 Overview: Developing

4-1 Define a simple data type called *Cities_All* in the Java Dictionary. Create some entries. The city values must have a prefix as follows:

EUR	European Cities
AMS	South American Cities
AMN	North American Cities
OCE	Oceanic Cities
ASA	Asian Cities
AFR	African Cities

4-2 Declare a value attribute *ctx_cityFrom* and *ctx_cityTo* (each of it type: Cities_All) in the context of the view controller.

4-3 Define two *InputField* UI elements with corresponding *Labels* for pickup location and drop down location. Define these elements in the pre-defined group UI element *grp_location*.

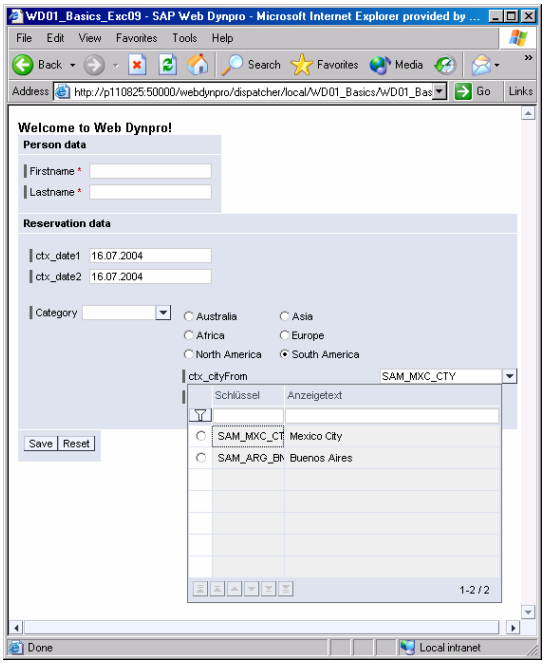
4-4 Define the data binding between the input fields and the corresponding context values.

4-5 Create the Web Dynpro application *WD01_Basics_GenericUI*.

5 Overview: Building, Deploying, and Running

Deploy and run the Web Dynpro application.

6 Optional: Dynamically fill value attributes



This exercise shows how to fill an individual value attribute dynamically with metadata at runtime:

You add a RadioButtonGroup UI element to pre-select the region of the cities.

After having selected the region, only the corresponding cities should be added to the value attributes.

6-1 Define two new data types in Java Dictionary

6-1-1 Define a simple data type called *Cities_Reg*.

6-1-2 Define a simple data type called *Region*. Create some entries. The region values must equal the city prefix from exercise 4-1:

EUR	Europe
AMS	South America
AMN	North America
OCE	Oceania
ASA	Asia
AFR	Africa

6-2 Define a new value attribute *ctx_region* in the context of the view controller (type *Region*).

6-3 Define a new radio button group UI element *RadioButtonGroupByKey* in the pre-defined group UI element *grp_location*. Bind the property *selectedKey* to the context field *ctx_region*.

6-4 Define a new action FilterCities. Bind this action to the property *onSelect* of the RadioButtonGroup. The corresponding Method will be processed after the user selects an entry from the RadioButtonGroup.

6-5 Declare a helper method *fillValueSet* to fill the value selector. This method shall be called at start time and after the selection of the RadioButtonGroup. At start time the value selector contains all cities.

After having selected a region from the RadioButtonGroup, the value selector should only contain cities of the selected region.

6-5-1 Declare the helper method *fillValueSet* (Parameter Boolean *initialize*).

6-5-2 Implement the Method *onActionFilterCities*. Call the method *fillValueSet(false)*.

6-5-3 Define a new value attribute *ctx_cityAll* (type: Cities_All) in the context of the view controller. This context field will be needed in the next step.

6-5-4 Implement the method *fillValueSet*.

- First get e reference an the context value attributes *ctx_cityAll*, *ctx_cityFrom* and *ctx_cityTo* using the method

```
wdContext.getNodeInfo()
          .getAttribute("<ctx_field>")
          .getModifiableSimpleType()
          .getSVServices()
          .getModifiableSimpleValueSet()
```

- Clear the Value Sets for the context value attributes *ctx_cityFrom* and *ctx_cityTo.*
- Loop over the value set for the context field *ctx_cityAll* and copy the values to the value sets for the context fields *ctx_cityFrom* and *ctx_cityTo.* Depending on the parameter *initialize*, only cities of a specified region shall be copied.

6-6 Update the method *wdDoInit()*. At start time the value selector should contain all cities. Therefore you have to call *fillValueSet(true)*.

6-7 Build, deploy and run the application