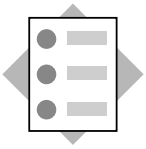
Generic UI Services

Contents:

- **Three Types of ValueHelp.**
- **Defining a static Simple Type containing a valueset.**
- **Simple Value Selector: Populating a DropDownByKey listbox with displaytexts.**
- **Extended Value Selector: Valuehelp for larger valuesets.**
- **Dynamic Type Modification: Modifying a value attribute´s datatype programmatically.**

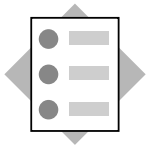
Generic UI Services: Objectives



After completing this lesson, you will be able to:

- **Explain the three types of ValueHelp.**
- **Use Static Valuesets.**
- **Use Dynamic Valuesets.**

Three Types of ValueHelp: Topic Objectives



After completing this topic, you will be able to:

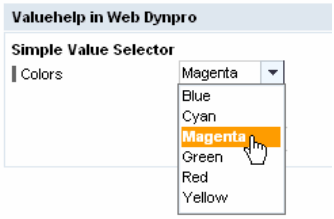
- **Explain the three types of ValueHelp.**

Three types of Valuehelp

Simple Value Selector

Based on a *DropDownByKey* UI Element bound to a context value attribute of type "Simple Type" containing a valueset

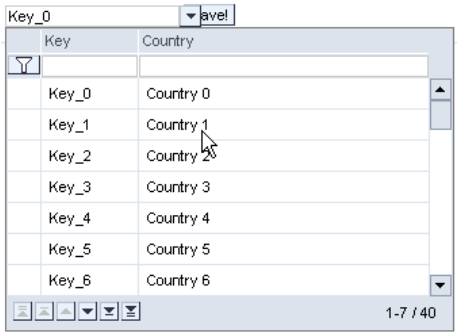
Used for small valuesets (less than 30 values)



Extended Value Selector

Based on an *Input Field* UI Element bound to a context value attribute of type "Simple Type" containing a valueset

Both types of valuehelp can be based on dynamically modified datatypes



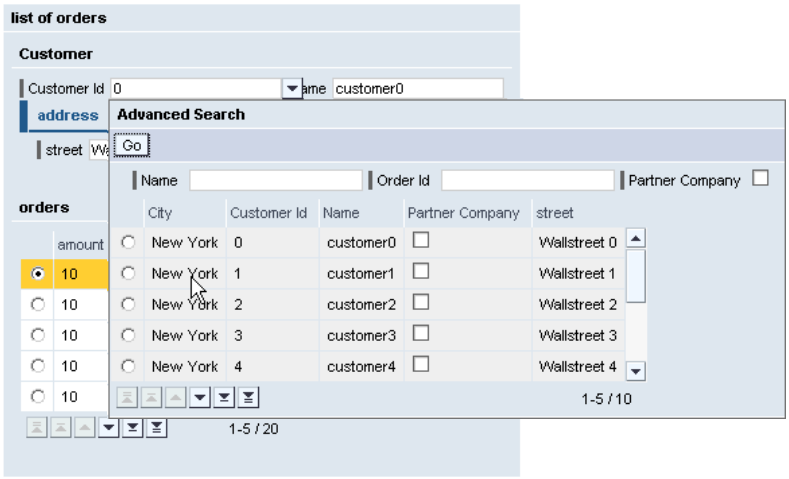
- **Generic UI Services**
  The Web Dynpro runtime environment provides generic UI services for the application development. They enable you to easily create a ValueHelp for constants – for example, for countries or zip codes.
- **Simple Value Selector (SVS)**
  You can use the simple value selector as a dropdown list box to display a set of constants when you bind a DropdownByKey UI element to a value attribute of the type Simple Type. The simple value selector is especially useful for small sets of constant values (up to 30 entries).
- **Extended Value Selector (EVS)**
  If the dropdown list box is too long due to the large number of constant values in a simple data type Simple Type, you should use the extended value selector. This value selector can display a large set of constants in a special dialog box below a regular input field. The extended value selector also provides a sort function and search function. The main concept of this value help is to bind constants to an input field or a DropdownByKey UI element. The constants are contained in a simple data type Simple Type and the value attributes have the same type. With the exception of the possible dynamic data type modifications of a value attribute at runtime, the application developer must only implement the declarations involved.

Three types of Valuehelp (2)

Object Value Selector

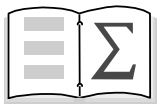
Advanced Search functionality

Based on declarative / programmatic approach



- **Object Value Selector (OVS)**
  In many application scenarios an additional type of valuehelp is needed for searching objects instead of values. Think of searching for a Airline ID. For finding this ID you want to enter some related data like departure and arrival airport or the flight date in a search form. The search results (matching flight objects) are displayed in a table and after selecting a flight the airplane ID (or other values) is (are) automatically transferred to the corresponding input field(s). For this purpose Web Dynpro provides a third type of generic valuehelp service called OVS (Object Value Selector).
  In contrast to SVS and EVS the Object Value Selector is not completely based on a declarative approach. For embedding this sophisticated valuehelp into your Web Dynpro application you have to implement some lines of code in an associated OVS custom controller. As a trade-off for your programmatic efforts the Web Dynpro Runtime automatically renders a generic OVS UI.
  This user interface is based on a special OVS core component belonging to the Web Dynpro Java Runtime Environment.

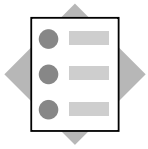
Three Types of ValueHelp: Topic Summary



You should now be able to:

- **Explain the three types of ValueHelp.**

Static versus Dynamic Valuesets: Topic Objectives

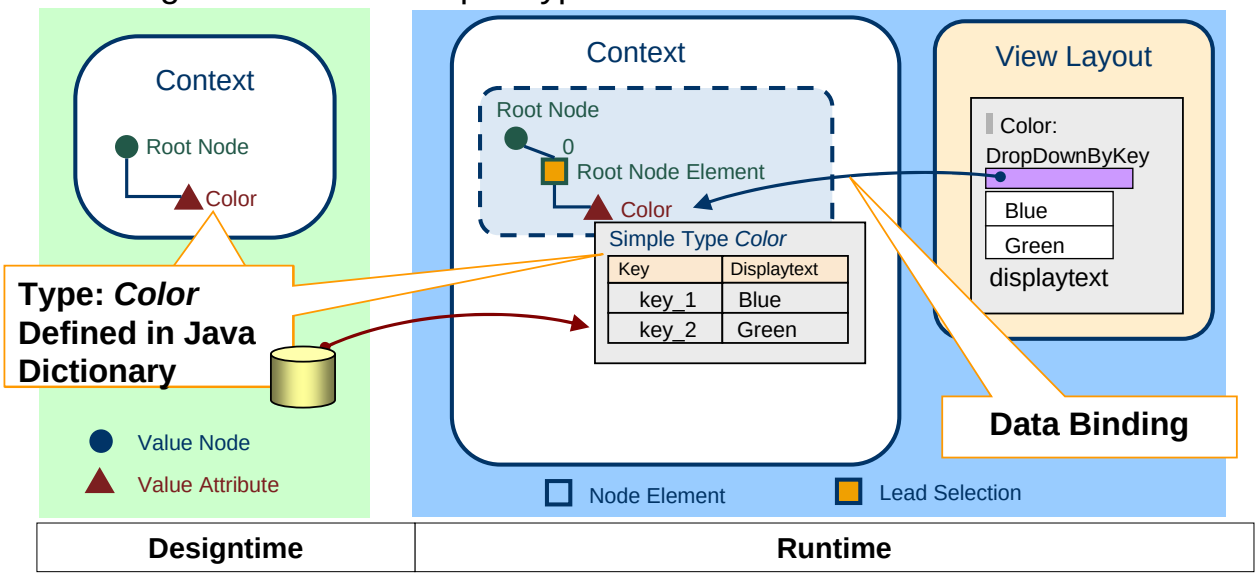


After completing this topic, you will be able to:

- **Use Static Valuesets.**
- **Use Dynamic Valuesets.**

Simple Value Selector with Static Valuesets

- DropDownByKey UI element property *selectedKey* is bound to context value attribute of type Simple Type (Java Dictionary).
- The dropdown listbox is automatically populated with the displaytexts being stored in the Simple Type´s metadata.



■ **How the Generic Valuehelp Service Works**

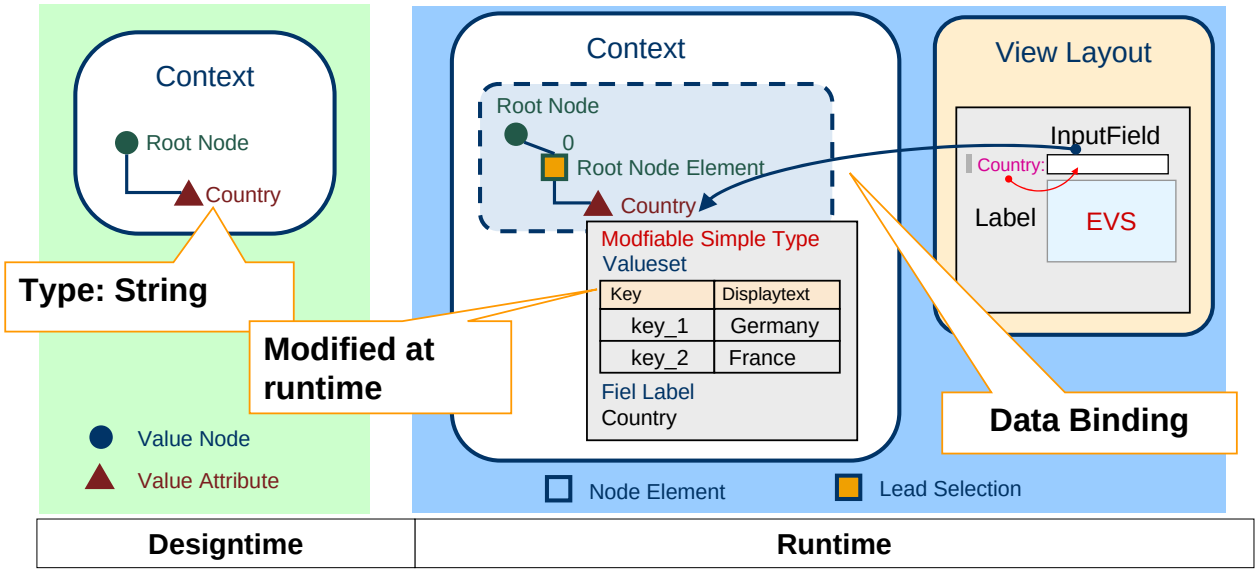
At design time, a simple data type *Simple Type* is assigned to a value attribute that is contained in the Java Dictionary. If the **selectedKey** property of a *DropDownByKey* UI element is bound to this value attribute, the dropdown list box – that is, the simple value selector – is automatically filled with entries in the view layout at runtime. These entries are stored in the data type of the value attribute. The value set is a list of key value pairs. Language-independent key values are always used in the source text, whereas the language dependent display texts are displayed in the dropdown list box.

Note that there is no data binding in the view context for the values to be displayed. Only the selected value is bound, all other information is provided by the data type of the value attribute.

This is an easy way to display metadata of a value attribute (type information of a *Simple Type*, such as constants or label text) on the user interface. In this example, the data type of the value attribute is statically defined at runtime.

Extended Value Selector with Dynamic Valuesets

- For representing valuesets in a context a value attribute can be typed using a dynamically modified datatype holding the valueset in its metadata.



Dynamic Valuesets

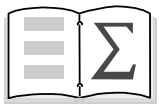
If the valueset to be displayed is not existing at design time, it has to be dynamically populated at runtime. For this reason the Programming Model API for Web Dynpro applications makes it possible to **modify the given datatype** of a single context-attribute

Dynamically Defining a Modifiable Simple Datatype

```
public void wdDoInit() {
  //@@begin wdDoInit()
  // Modify SimpleType of context value attribute named Country
  IWDAttributeInfo attributeInfo =
    wdContext.getNodeInfo().getAttribute("Country");
  ISimpleTypeModifiable countryType =
    attributeInfo.getModifiableSimpleType();
  countryType.setFieldLabel("Country");
  IModifiableSimpleValueSet valueSet =
    automakerType.getSVServices().getModifiableSimpeValueSet();
  valueSet.put("DE", "Germany");
  valueSet.put("GB", "Great Britain");
  valueSet.put("US", "United States");
  valueSet.put("ALL", "All");
  wdContext.currentContextElement().setCountry("ALL");
  //@@end
}
```


- **Example: Modifying a context value attribute's datatype**
  In the example an extended value selector is used where the set of constants is not statically available at design time but only at runtime.
  The initialization of the view controller dynamically modifies the statically declared data type of the value attribute *Country* at runtime.
  Note that the modification of the data type *string* only affects one value attribute even if the same type is used multiple times somewhere else.
  In addition to the set of constants, the FieldLabel attribute is set, which is then automatically displayed in the *Label* UI element in front of the *Country* input field. This requires the previous assignment of the name of the *Country* input field **InputField** to the **labelFor** property of the *Label* UI element at design time. Since this input field is bound to a value attribute, you can determine the label text to be displayed at runtime using the value attribute metadata of the modified data type *Simple Type*.

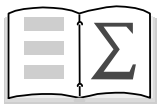
Static versus Dynamic Valuesets: Topic Summary



You should now be able to:

- **Use Static Valuesets.**
- **Use Dynamic Valuesets.**

Generic UI Services: Summary



You should now be able to:

- **Explain the three types of ValueHelp.**
- **Use Static Valuesets.**
- **Use Dynamic Valuesets.**