Reuse Java Dictionary Types in Web Dynpro DCs

SAP NetWeaver 04

Copyright

Reuse Java Dictionary Types in Web Dynpro DCs

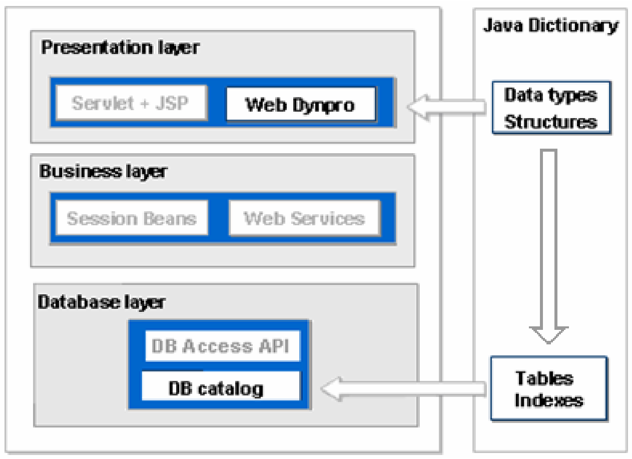
Abstract

Within SAP NetWeaver 04 there are various architecture and design frameworks that enable the efficient and consistent development of software according to the “write once, reuse everywhere” paradigm: Defining application objects once and reuse them extensively. Among others, the Java Dictionary framework (Dictionary) applied in conjunction with the SAP NetWeaver Development Infrastructure (NWDI) can help you avoid redundant data type definitions.

Our tutorial demonstrates how to apply data types defined in a Development Component (DC) of type Dictionary as types of database table columns and, on the other hand, as the data types of UI elements within Web Dynpro for Java views. Consider the following scenario:

- ✓ Within Dictionary DCs, data domain experts
    - o Develop application specific data types.
    - o Apply them as types of database table columns.
    - o Check the types into the Design Time Repository server (DTR) of NWDI.
- ✓ Subsequently, Web Dynpro skilled developers can check them out and reuse consistently within Web Dynpro DCs.

This way Dictionary DCs can act as a type repository of the database layer and presentation layer of your, possibly complex, Java application, as you can view in the picture below. If you change a Dictionary type, all the objects that use it are established upon generation. The objects that are found are automatically adjusted to include the change.



Objectives and Target Group

This guide is aimed at Java developers building enterprise systems on top of the SAP NetWeaver platform. Especially:

- Web Dynpro UI programmer can learn how to use simple types from the Java Dictionary to define the data types of UI elements.
- Database programming specialists can learn how to use simple types from the Java Dictionary to define the data types of database table columns.

Prerequisites

You are a Java developer with a basic knowledge in applying the following SAP NetWeaver frameworks:

- SAP NetWeaver Development Infrastructure (NWDI)
- Web Dynpro for Java (hereafter referred to simply as Web Dynpro or just WD) and/or the Java Dictionary.

Limitation

During the build process of the importing Web Dynpro DC, Java Dictionary types are replicated into the Web Dynpro DC, what can cause a negative impact on the performance of the SAP NetWeaver Developer Studio.

As of the Support Package Stack 10 of SAP NetWeaver 04, a similar problem has been solved if you import data types defined inside a Web Dynpro DC. Defining data types inside a Web Dynpro DC will therefore offer better performance of the build process in the SAP NetWeaver Developer Studio.

Roadmap

We will go through the following successive development steps:

1. Obtain NWDI development configuration
2. Create Dictionary DC and define dictionary data type (dictionary type)
3. Apply dictionary type as data type of database table column
4. Make dictionary type visible to another DCs
5. Create Web Dynpro DC and define UI field
6. Import Dictionary Type
7. Apply dictionary type as data type of UI field
8. Test Web Dynpro application

At various points in the text, you will find background information to explain the underlying concepts.

Steps

Obtain NWDI Development Configuration

In the Development Configurations perspective of the SAP NetWeaver Developer Studio (NWDS), we log on to our Development Infrastructure Landscape and import our Development Configuration.

The Development Configuration determines among other things, the development track, the Design Time Repository server and the Name Server.

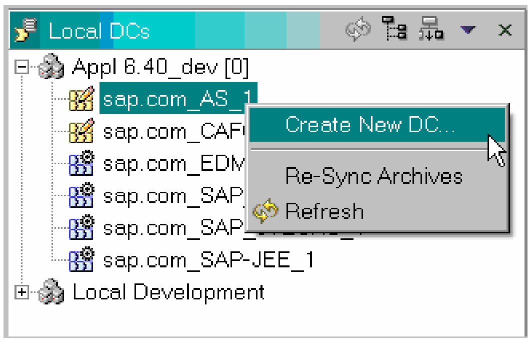
- A **development track** is a separate production line for a certain release of a software component. Among other things, it defines a workspace folder holding your development projects.
- To share development objects, all team members store (or “check in”) application sources within a **Design Time Repository** (DTR) server.
- Globally distributed software development requires a conflict-free method of creating names for development projects, database tables, web context names and other

application objects. The **Name Server** makes sure that the names we apply belong to our own namespace and that their subsequent reuse is not possible.

Create Dictionary DC

In the context of NWDI, you always create Dictionary projects as Development Components of type Dictionary.

We would like to use a new Dictionary DC so let's create one:

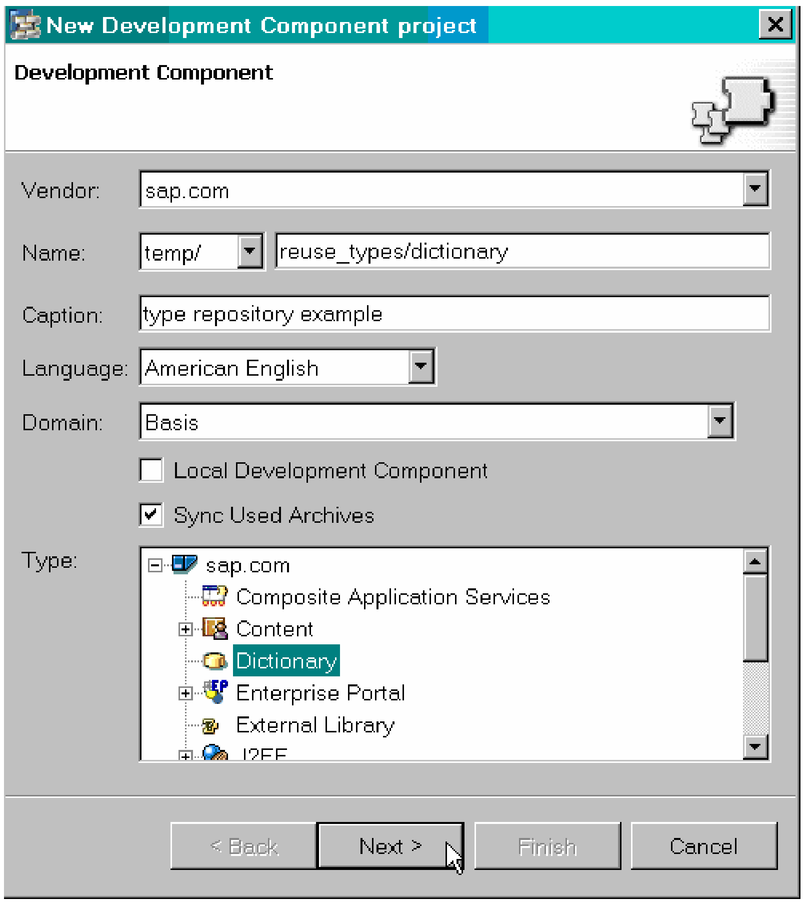


Important properties of the new DC are:

Name:
The Name Sever is listing available name prefixes (namespaces). We select the one well suited for examples and tests. In addition, we enter a name suffix. The Name Server will reserve the whole name and no one will be able to re-apply it.

Type: Dictionary

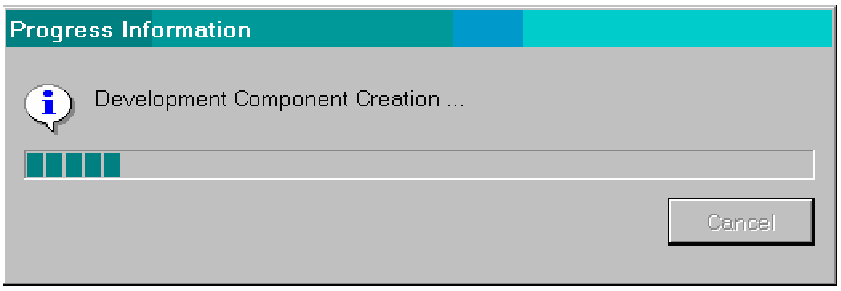
Here is a screenshot of the definition of our Dictionary DC:



New Development Component project

Development Component

Vendor:	sap.com
Name:	temp/ reuse_types/dictionary
Caption:	type repository example
Language:	American English
Domain:	Basis
	☐ Local Development Component
	☑ Sync Used Archives
Type:	sap.com: Composite Application Services, Content, **Dictionary**, Enterprise Portal, External Library, J2EE



Progress Information

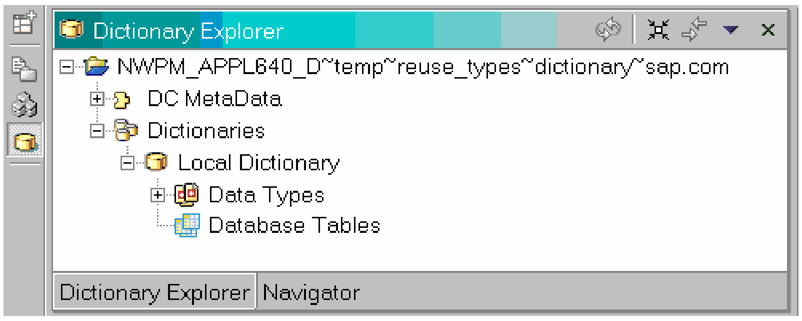
Development Component Creation ...

Finally, a Dictionary DC project opens ready to edit.

In the Dictionary Explorer View

The Dictionary Explorer view of the Dictionary Perspective contains Dictionary editors with which you can define data types and database tables.

- The folder Dictionaries stores data types and definitions of database tables.
- The folder DC MetaData stores NWDI related DC data.



The NWDS makes Dictionary type editors available to several kinds of projects:

- Dictionary DCs
- Web Dynpro DCs
- Local Dictionary and Web Dynpro projects, applied without any connection to **the** NWDI

In each case, the functionality and handling of type editors is almost the same.

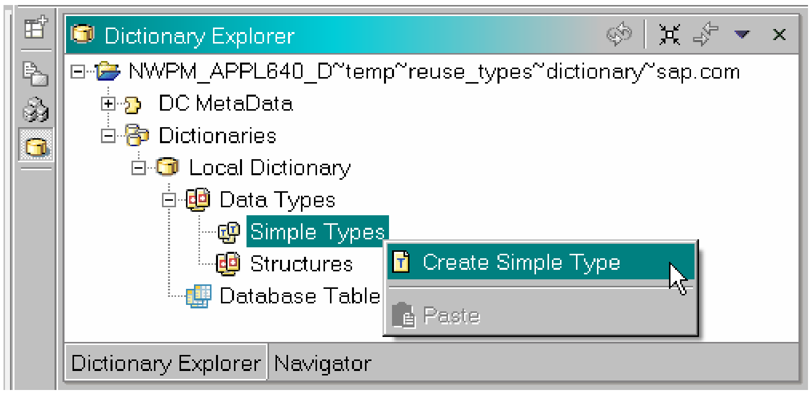
When using Dictionary and Web Dynpro DCs, the NWDS always tracks your modifications and forces you to add each one to an activity.

Define Dictionary Data Type

On top of predefined JDBC-like built-in data types of the Java Dictionary, you can define more complex, application specific data types: these are known as simple types.

It is good practice always to implement your application specific data types as simple types in the Java Dictionary.

To create a new simple type, right click the Simple Types node:

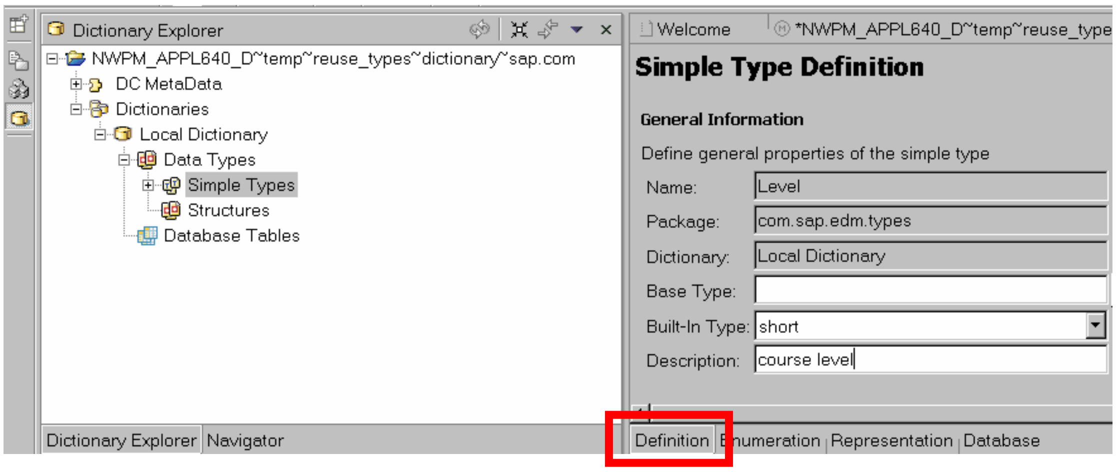


We call our new simple type “Level”. This simple type is intended to quantify the degree of skills necessary to attend an educational course or to work on a specific project.

If you create a new simple type (or double click on an existing one), the type editor opens. Let’s go through the tabs Definition, Enumeration, Representation and Database and set attributes of the type Level. Later on you can see that the table editor of Java Dictionary and the Web Dynpro framework will adopt these settings.

Definition Tab:
Here you define basic settings for the type like

- Name: Level
- Package: com.sap.edm.types
- Built-in type: short



Simple Type Definition

General Information

Define general properties of the simple type

Name:	Level
Package:	com.sap.edm.types
Dictionary:	Local Dictionary
Base Type:	
Built-In Type:	short
Description:	course level

You may possibly be wondering why we do not apply common Java types here. Here is a short explanation: One of the main goals of SAP NetWeaver is portability across the database platforms that SAP supports. For this reason in the Dictionary editor, you make use of the Dictionary's built-in types. Actually, built-in types simply imitate JDBC/Java types. The Dictionary framework immediately maps them to proper JDBC types (database tables) and Java types (Web Dynpro pages), as we will demonstrate later.

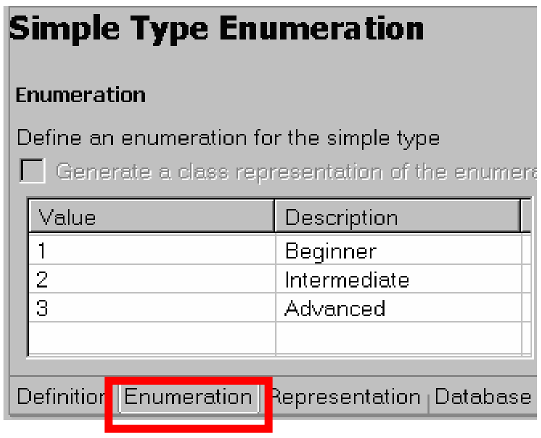
It is important to know type mapping rules:

Dictionary **built-in** type ⇦⇨ **JDBC** type ⇦⇨ **Java** type.

You can find the mapping rules in the SAP NetWeaver documentation.

Next, change to the Enumeration tab if you wish to restrict the values permitted for the simple type to a hard coded list. If a Web Dynpro UI elements is based upon a Dictionary Simple Type that has a defined enumeration, then when the Web Dynpro screen is rendered, the enumeration values will automatically appear either as a drop down list (known as a Simple Value Selector), or as a pop-up table with filtering options (known as an Extended Value Selector)

The following screen shot shows the only values permitted for fields based on the Simple Type Level:


Simple Type Enumeration

Enumeration

Define an enumeration for the simple type

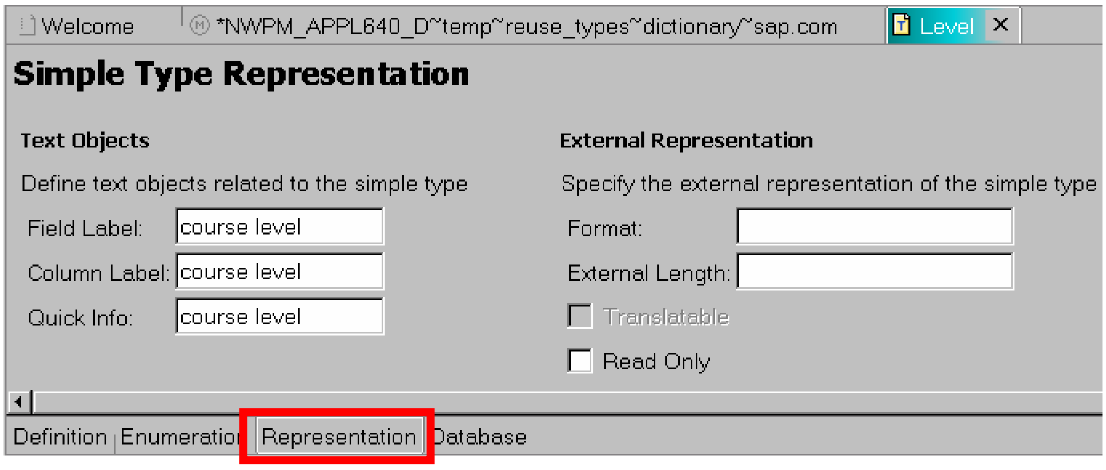
☐ Generate a class representation of the enumera

Value	Description
1	Beginner
2	Intermediate
3	Advanced

In the tab Representation you can enter additional descriptions like

- Field Label
- Column Label
- Quick Info

Web Dynpro will display these texts for fields of type Level. We enter “course level” three times:



Simple Type Representation

Text Objects

Define text objects related to the simple type

Field Label:	course level
Column Label:	course level
Quick Info:	course level

External Representation

Specify the external representation of the simple type

Format:	
External Length:	

☐ Translatable

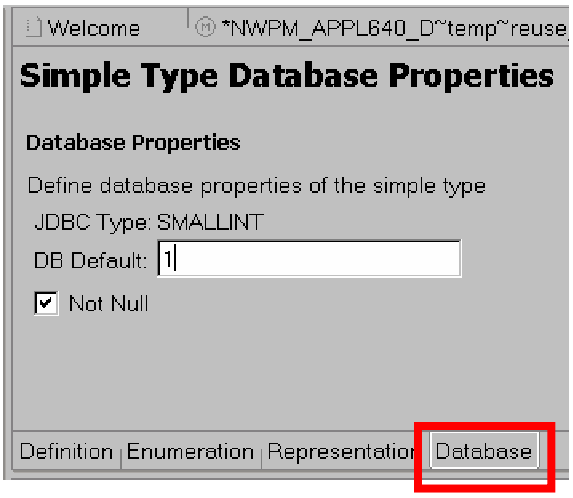
☐ Read Only

(Remaining settings, also available here, are not related to our example.)

Finally, in the tab Database, you can decide whether database table fields of this type are permitted to be null. You can also set a default value for them. We decide that the attribute level has to be set to a default value, and that the field cannot be null. Therefore we select:

DB Default: 1

Not Null: Checked


Welcome | *NWPM_APPL640_D~temp~reuse

Simple Type Database Properties

Database Properties

Define database properties of the simple type

JDBC Type: SMALLINT

DB Default: 1

[✓] Not Null

Definition | Enumeration | Representation | Database

Dictionary Type as Data Type of Database Table Column

Now we can make a straightforward application of the simple type Level as a database table column type. In this way, you ensure that the database fields are consistent with centrally defined application types.

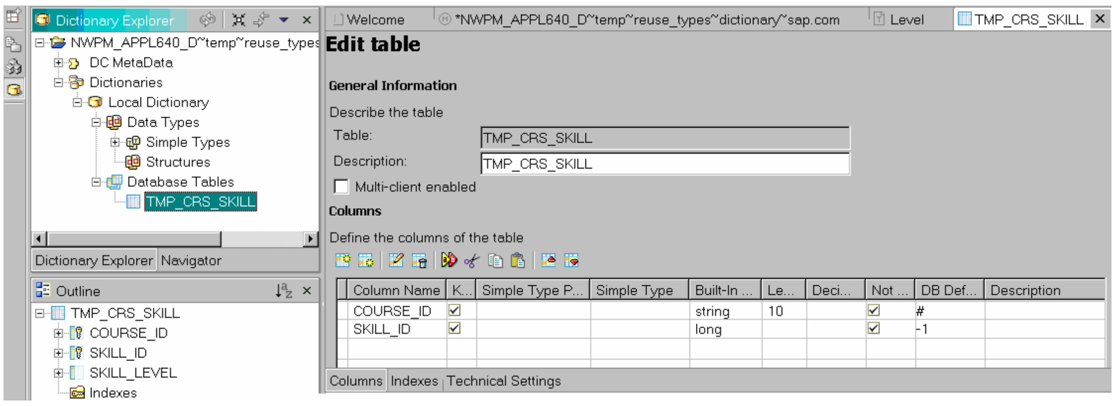
Within Dictionary (DC) projects you define database tables “offline”, without being connected to any database instance. If you deploy the resulting sda archive to the server, the Dictionary framework automatically creates (updates) corresponding database tables in the underlying system database.

Dictionary table definitions are portable. Therefore, they work immediately on any DBMS supported by the Java stack of SAP NetWeaver. Moreover, several editors of the Developer Studio check if your database access code and the related dictionary tables match. This early consistency check improves the robustness of your application and helps shorten the development process.

Our database table belongs to the same Dictionary DC as the simple type Level. Alternatively, you could import the type Level into another Dictionary DC and reuse it there[1]. (We show the steps required for the import later on, while importing the type into the Web Dynpro DC.)

View our definition of a database table called TMP_EDM_SKILL. The table is a join table between educational courses and skills each course is intended to address.

[1] The imported types are replicated into the embedding Dictionary DCs, which can cause a negative impact on the performance of the SAP NetWeaver Developer Studio. This problem has been solved for Web Dynpro DCs starting with the Support Package Stack 10 of SAP NetWeaver 04, but still exists for DCs of type Dictionary.



Edit table

General Information

Describe the table

Table:	TMP_CRS_SKILL
Description:	TMP_CRS_SKILL

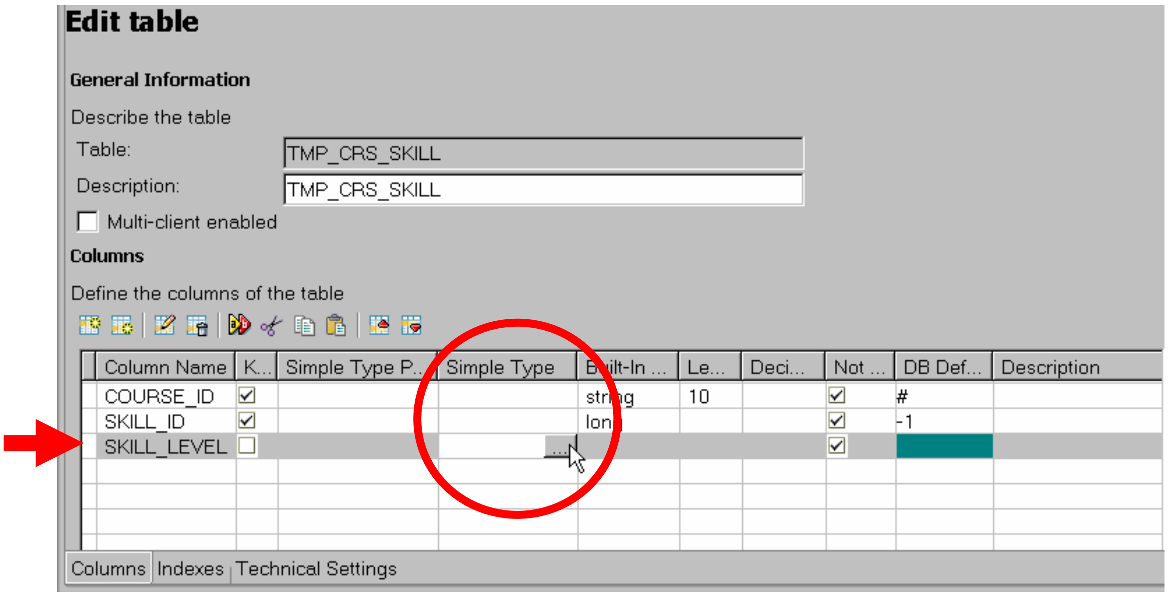
☐ Multi-client enabled

Columns

Define the columns of the table

Column Name	K...	Simple Type P...	Simple Type	Built-In ...	Le...	Deci...	Not ...	DB Def...	Description
COURSE_ID	✓			string	10		✓	#	
SKILL_ID	✓			long			✓	-1	

To facilitate a fine-grained skill specification we decide to add a column SKILL_LEVEL to the table. You may be thinking at this point that a built-in type from the Java Dictionary could be used to define the type of column. However, we are concerned about the type consistency in the whole application and therefore apply our simple type Level as the type of the column:



Edit table

General Information

Describe the table

Table:	TMP_CRS_SKILL
Description:	TMP_CRS_SKILL

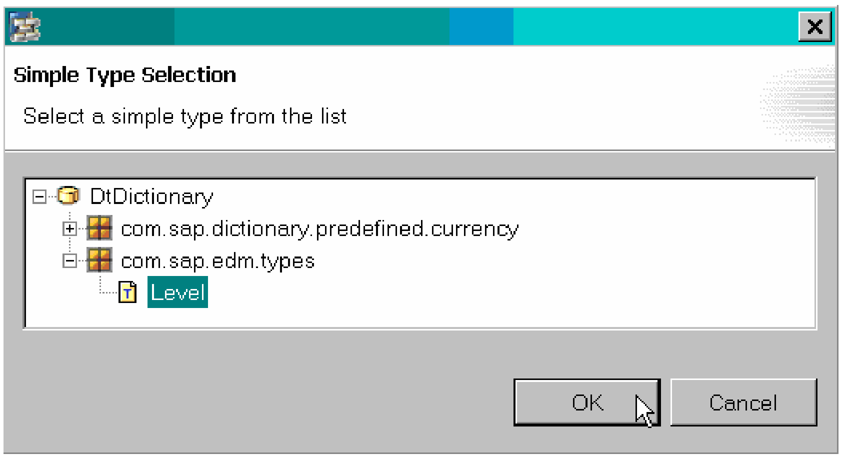
☐ Multi-client enabled

Columns

Define the columns of the table

Column Name	K...	Simple Type P...	Simple Type	Built-In ...	Le...	Deci...	Not ...	DB Def...	Description
COURSE_ID	✓			string	10		✓	#	
SKILL_ID	✓			long			✓	-1	
SKILL_LEVEL	☐		...				✓		

Very easy: Select the type Level from the list …



Simple Type Selection

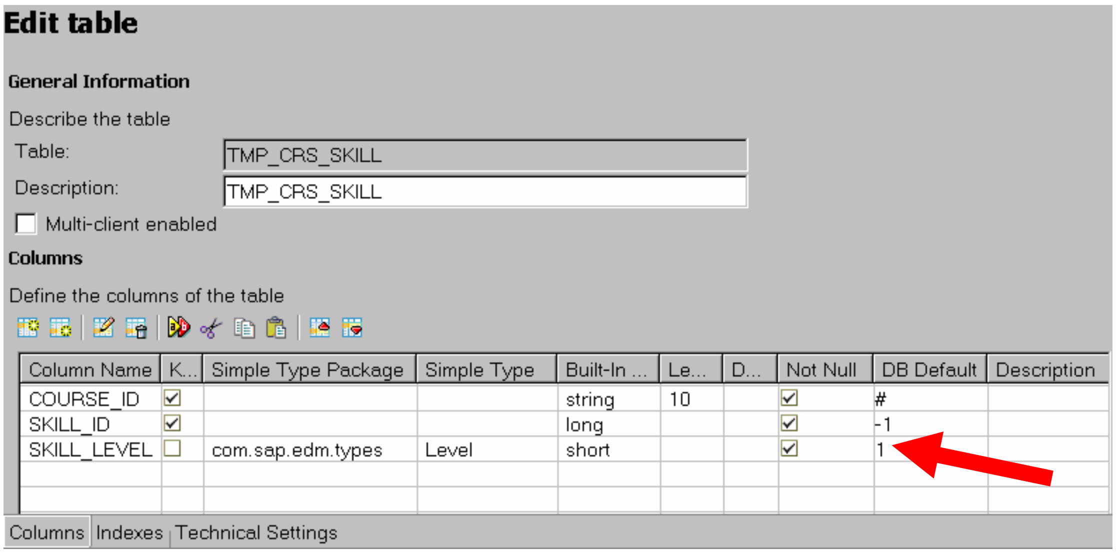
Select a simple type from the list

- DtDictionary
  - com.sap.dictionary.predefined.currency
  - com.sap.edm.types
    - Level

OK, fine, as you can see below, the table field SKILL_LEVEL has automatically inherited several settings:

- ✓ name and package
- ✓ built-in type
- ✓ not null
- ✓ db default

from the simple type Level:



Edit table

General Information

Describe the table

Table: TMP_CRS_SKILL

Description: TMP_CRS_SKILL

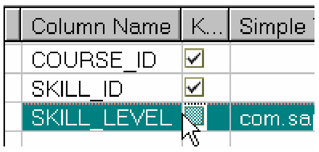
☐ Multi-client enabled

Columns

Define the columns of the table

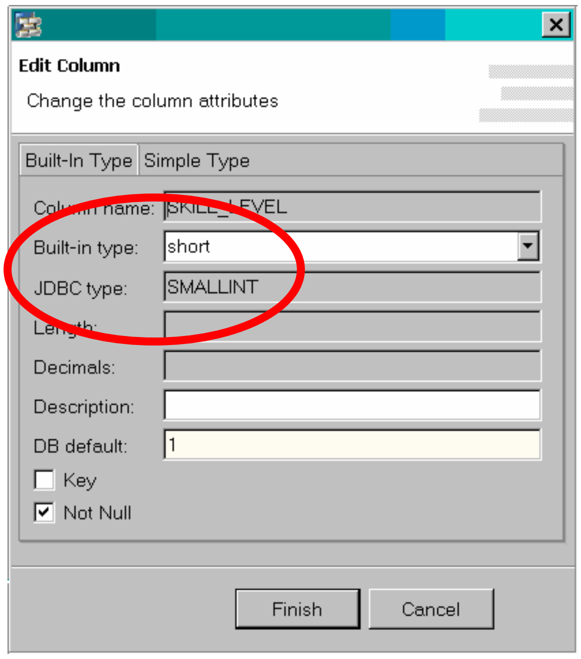
Column Name	K...	Simple Type Package	Simple Type	Built-In ...	Le...	D...	Not Null	DB Default	Description
COURSE_ID	☑			string	10		☑	#	
SKILL_ID	☑			long			☑	-1	
SKILL_LEVEL	☐	com.sap.edm.types	Level	short			☑	1	

Are you interested in knowing what the JDBC type of the column is? Then double click on the Column Name field and you can view all the column attributes:



Column Name	K...	Simple
COURSE_ID	☑	
SKILL_ID	☑	
SKILL_LEVEL	▨	com.sa

The built-in type short maps to the JDBC type SMALLINT in the database layer:



Edit Column

Change the column attributes

Column name: SKILL_LEVEL

Built-in type: short

JDBC type: SMALLINT

Length:

Decimals:

Description:

DB default: 1

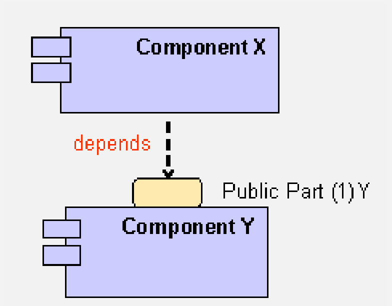
☐ Key

☑ Not Null

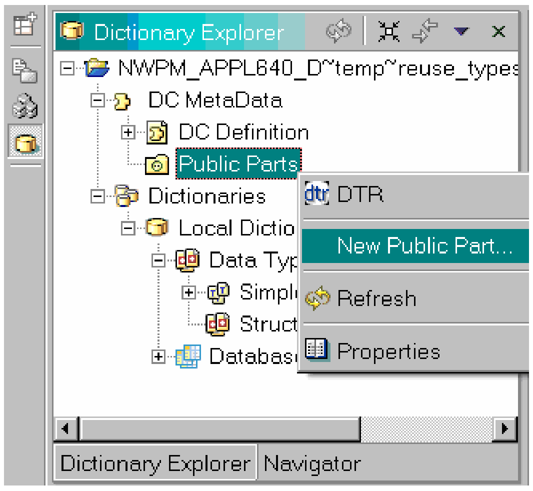
Make Dictionary Type Visible to Another DCs

Now we would like to apply our type as a field type in a Web Dynpro View, too. The first step is to add the simple type Level to the public part of the Dictionary DC.

Development components behave like a “black box” since their content is not visible from the outside. The interfaces of a DC to the outside world are known as **public parts**. A public part lists development objects offered to other DCs, as depicted below:

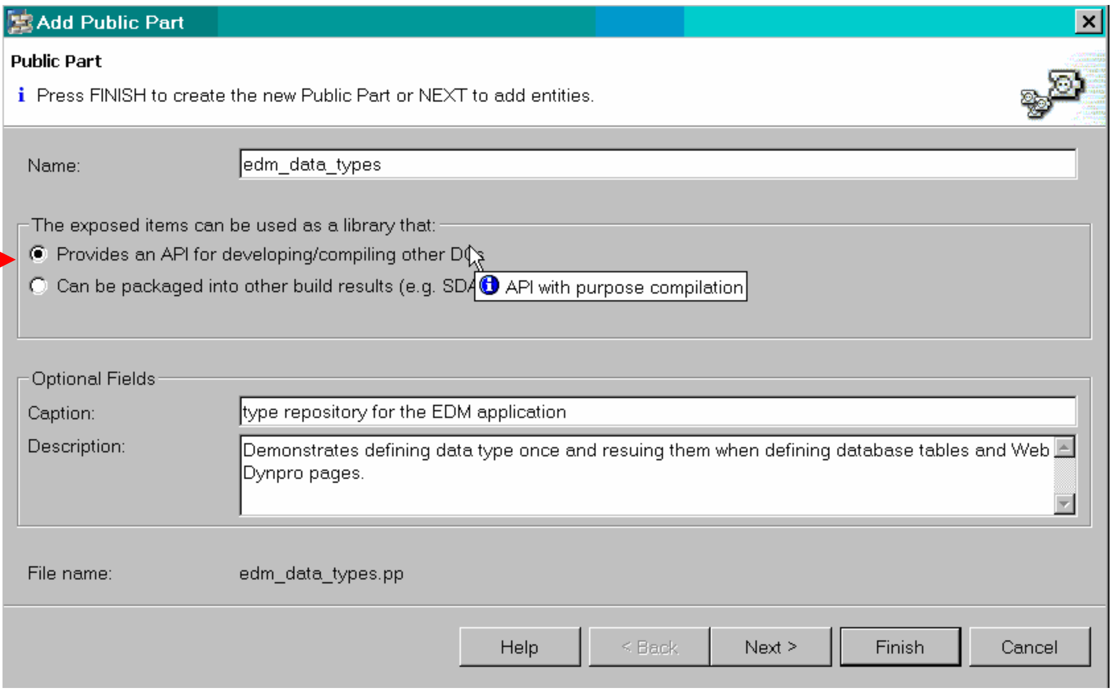


At the moment, our Dictionary DC does not have any public parts, so we must create one:



In the upcoming dialog

- Enter a name of the public part
- Select the “Provides an API for developing/compiling other DCs” option


Add Public Part

Public Part

Press FINISH to create the new Public Part or NEXT to add entities.

Name: edm_data_types

The exposed items can be used as a library that:

- [✓ Provides an API for developing/compiling other DC]
- Can be packaged into other build results (e.g. SDA

API with purpose compilation

Optional Fields

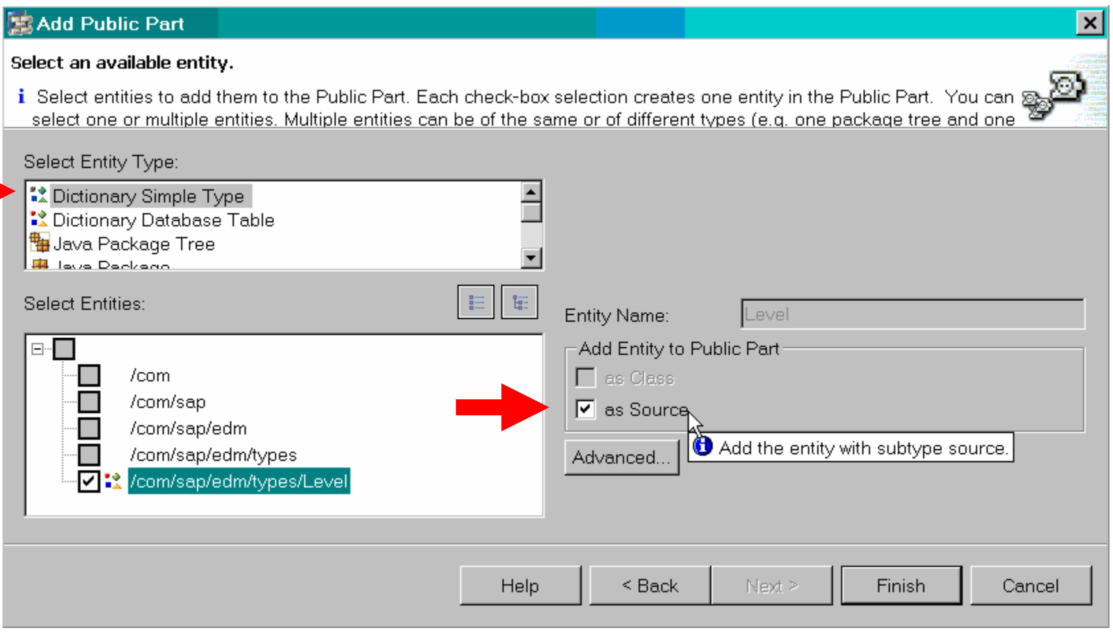
Caption: type repository for the EDM application

Description: Demonstrates defining data type once and resuing them when defining database tables and Web Dynpro pages.

File name: edm_data_types.pp

Finally, we can add our type to the newly created public part. We navigate to the simple type Level and select the proper options:

- Entity Type: Dictionary Simple Type
- Add Entity to Public Part: as Source


Add Public Part

Select an available entity.

Select entities to add them to the Public Part. Each check-box selection creates one entity in the Public Part. You can select one or multiple entities. Multiple entities can be of the same or of different types (e.g. one package tree and one

Select Entity Type:

- Dictionary Simple Type
- Dictionary Database Table
- Java Package Tree
- Java Package

Select Entities:

- /com
- /com/sap
- /com/sap/edm
- /com/sap/edm/types
- [✓ /com/sap/edm/types/Level]

Entity Name: Level

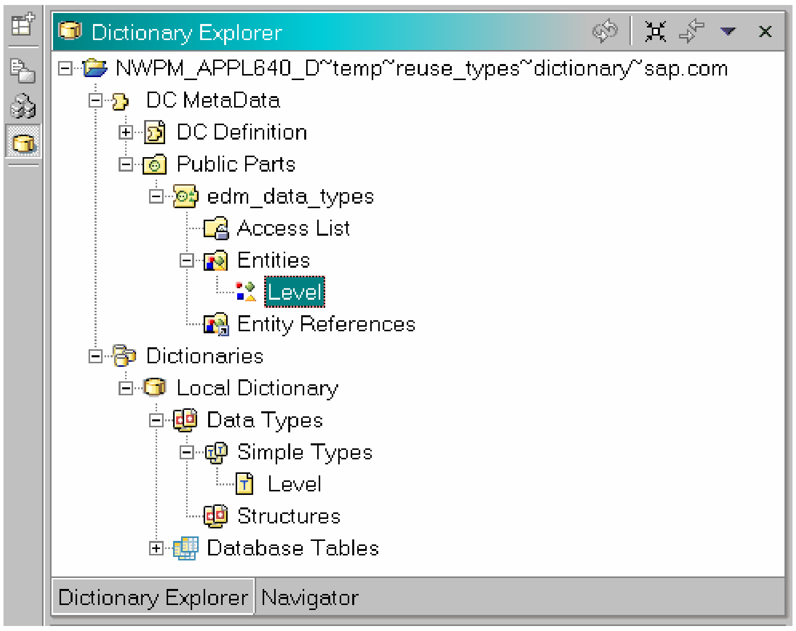
Add Entity to Public Part

- as Class
- [✓ as Source]

Add the entity with subtype source.

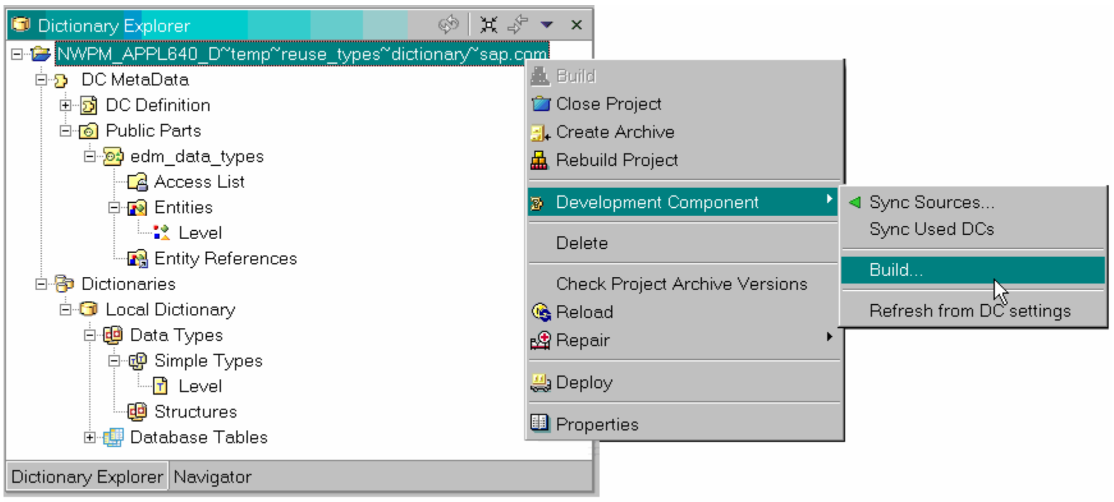
Press Finish.

Now the simple type Level appears as an Entity in a public part of our Dictionary DC:



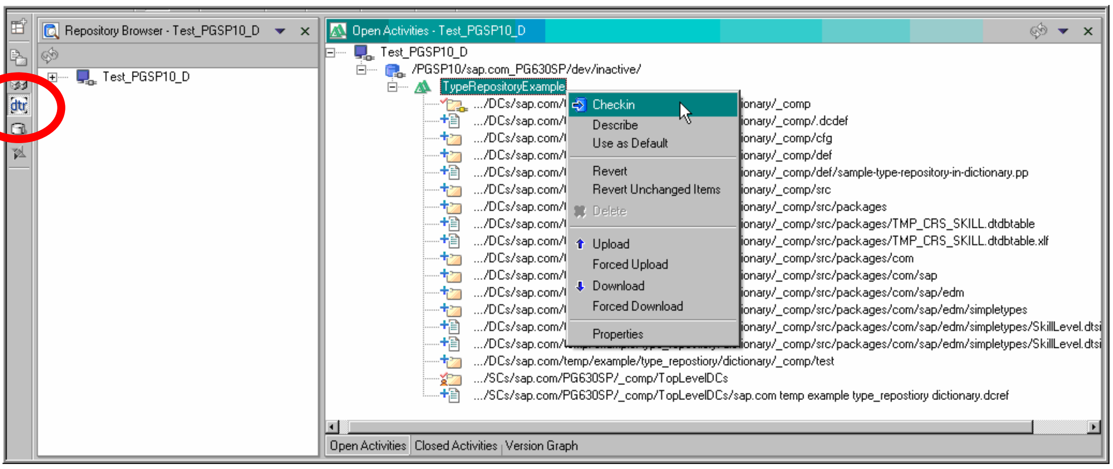
To round off the type publishing we execute the following NWDI related tasks:

- Build the DC



Development objects must be built before they can be deployed in a runtime system or before other DCs can use them. In general, whenever you change or add an object within a Public Part, you must perform a DC build in order to make it visible from the outside.

- Check the DC into the DTR repository.
  Check the whole DTR activity in:

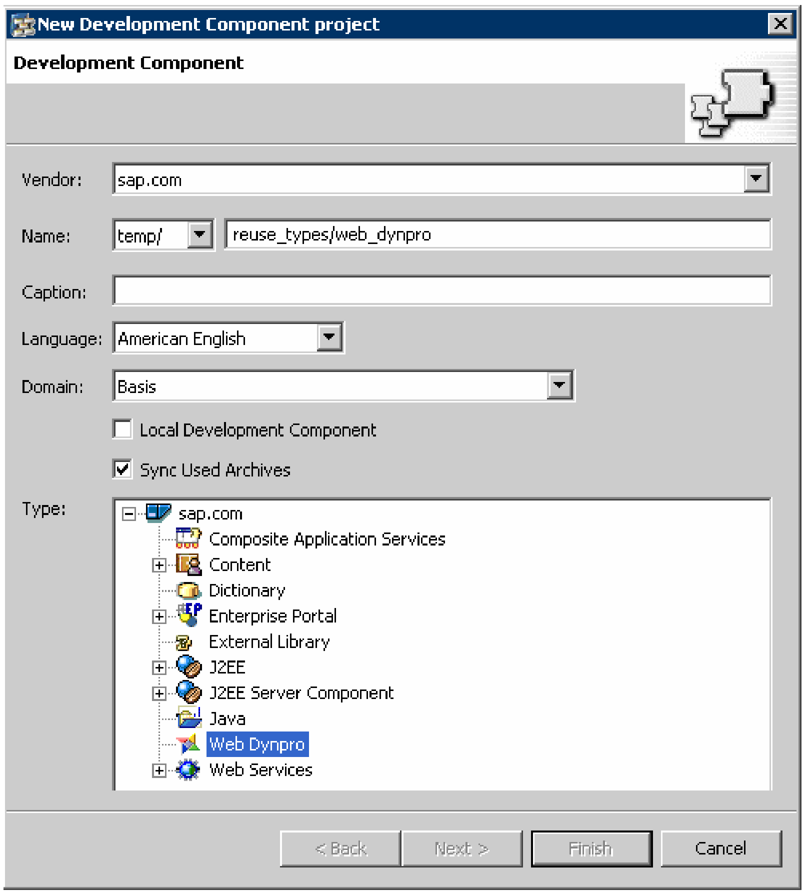


```
Test_PGSP10_D
  /PGSP10/sap.com_PG630SP/dev/inactive/
    TypeRepositoryExample
      .../DCs/sap.com/t        ionary/_comp
      .../DCs/sap.com/t        ionary/_comp/.dcdef
      .../DCs/sap.com/t        ionary/_comp/cfg
      .../DCs/sap.com/t        ionary/_comp/def
      .../DCs/sap.com/t        ionary/_comp/def/sample-type-repository-in-dictionary.pp
      .../DCs/sap.com/t        ionary/_comp/src
      .../DCs/sap.com/t        ionary/_comp/src/packages
      .../DCs/sap.com/t        ionary/_comp/src/packages/TMP_CRS_SKILL.dtdbtable
      .../DCs/sap.com/t        ionary/_comp/src/packages/TMP_CRS_SKILL.dtdbtable.xlf
      .../DCs/sap.com/t        ionary/_comp/src/packages/com
      .../DCs/sap.com/t        ionary/_comp/src/packages/com/sap
      .../DCs/sap.com/t        ionary/_comp/src/packages/com/sap/edm
      .../DCs/sap.com/t        ionary/_comp/src/packages/com/sap/edm/simpletypes
      .../DCs/sap.com/t        ionary/_comp/src/packages/com/sap/edm/simpletypes/SkillLevel.dtsi
      .../DCs/sap.com/t        ionary/_comp/src/packages/com/sap/edm/simpletypes/SkillLevel.dtsi
      .../DCs/sap.com/temp/example/type_repostiory/dictionary/_comp/test
      .../SCs/sap.com/PG630SP/_comp/TopLevelDCs
      .../SCs/sap.com/PG630SP/_comp/TopLevelDCs/sap.com temp example type_repostiory dictionary.dcref
```

Finally, the simple type Level is ready to be reused by other members of our development team.

Create Web Dynpro DC

We would now like to use the simple type Level as the definition of a Web Dynpro UI element. Therefore, the first task is to create a new Web Dynpro DC. This time, we select Web Dynpro as the type of the new Development Component:

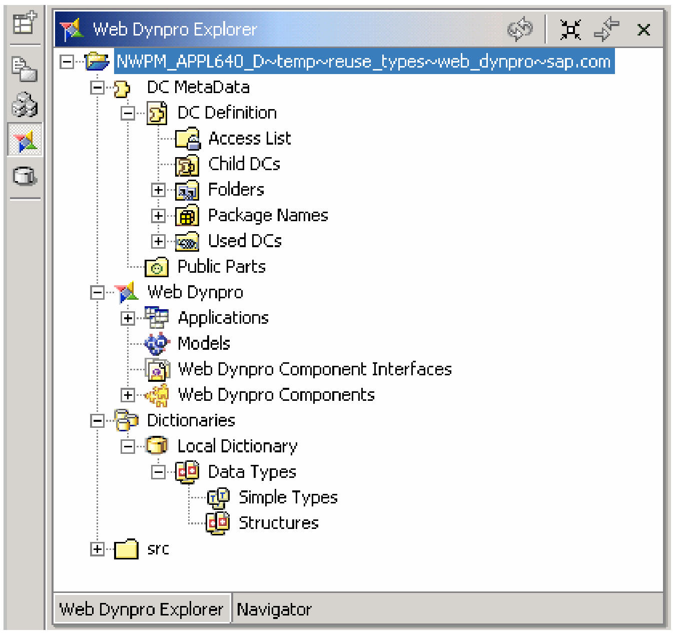


Vendor:	sap.com
Name:	temp/ reuse_types/web_dynpro
Caption:	
Language:	American English
Domain:	Basis
Local Development Component	☐
Sync Used Archives	✓
Type:	Web Dynpro

Now that the Web Dynpro DC has been created, you need to switch into the Web Dynpro Explorer Perspective in the NWDS, since it is only from within this perspective that you have access to the various Web Dynpro editors for defining Web Dynpro components, views, windows and UI elements.

Here is the Web Dynpro Explorer view of your DC. It holds Web Dynpro objects such as:

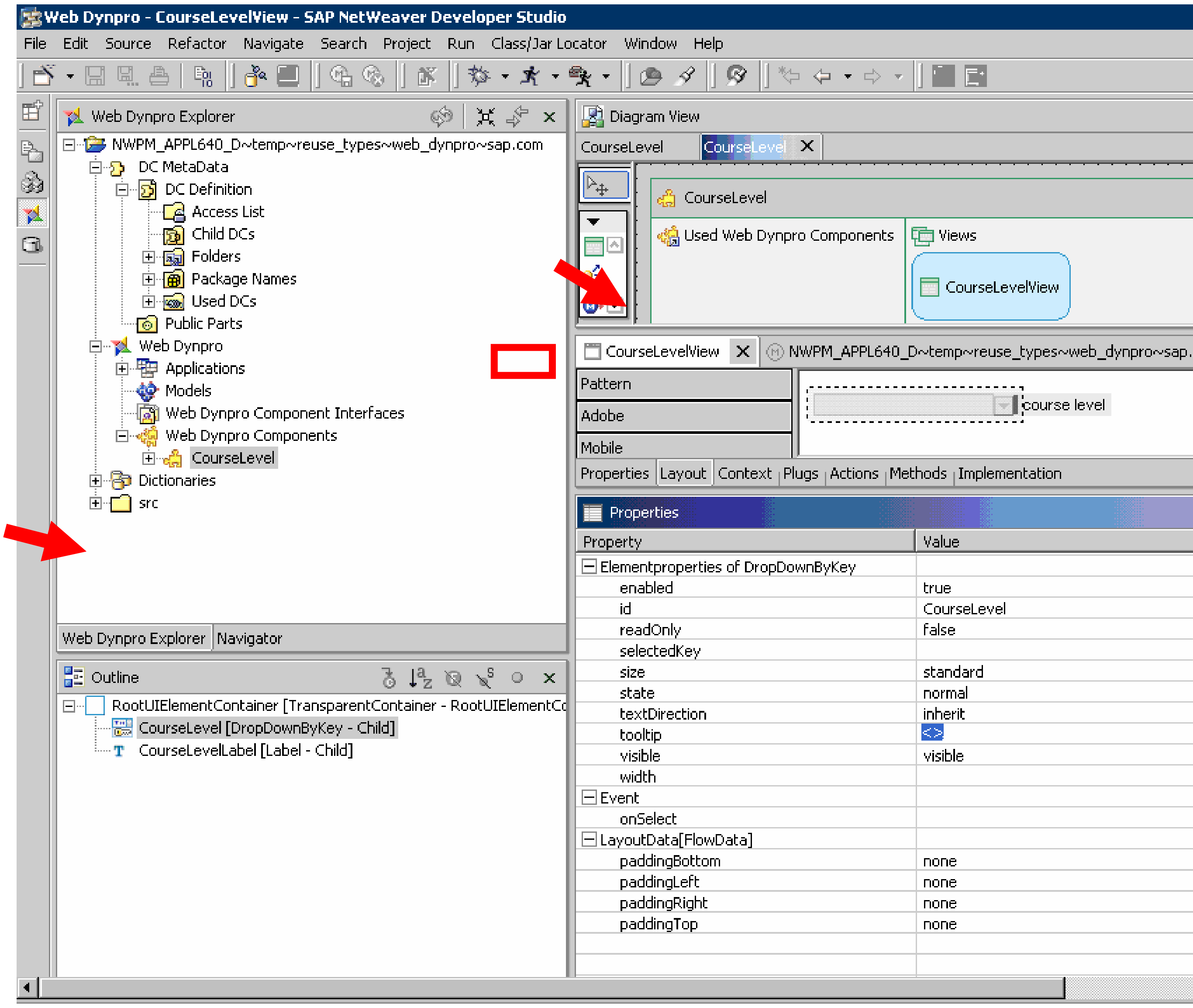
- The DC MetaData to store NWDI related DC data
- Dictionary information about the data types used in the DC.



As you can see in the picture above, data types you define in Web Dynpro projects are called simple types as in Dictionary projects. Actually, in both cases, simple types have equal behaviour (and you use the same type editors).

Define UI Field

Below you can view the definition of a Web Dynpro list box called CourseLevel:


Web Dynpro - CourseLevelView - SAP NetWeaver Developer Studio
File Edit Source Refactor Navigate Search Project Run Class/Jar Locator Window Help
Web Dynpro Explorer
Diagram View
CourseLevel
CourseLevel
Used Web Dynpro Components
Views
CourseLevelView
Pattern
Adobe
Mobile
course level
Properties Layout Context Plugs Actions Methods Implementation
Web Dynpro Explorer
Navigator
Outline

NWPM_APPL640_D~temp~reuse_types~web_dynpro~sap.com
- DC MetaData
  - DC Definition
    - Access List
    - Child DCs
    - Folders
    - Package Names
    - Used DCs
  - Public Parts
- Web Dynpro
  - Applications
  - Models
  - Web Dynpro Component Interfaces
  - Web Dynpro Components
    - CourseLevel
- Dictionaries
- src

CourseLevelView | NWPM_APPL640_D~temp~reuse_types~web_dynpro~sap.

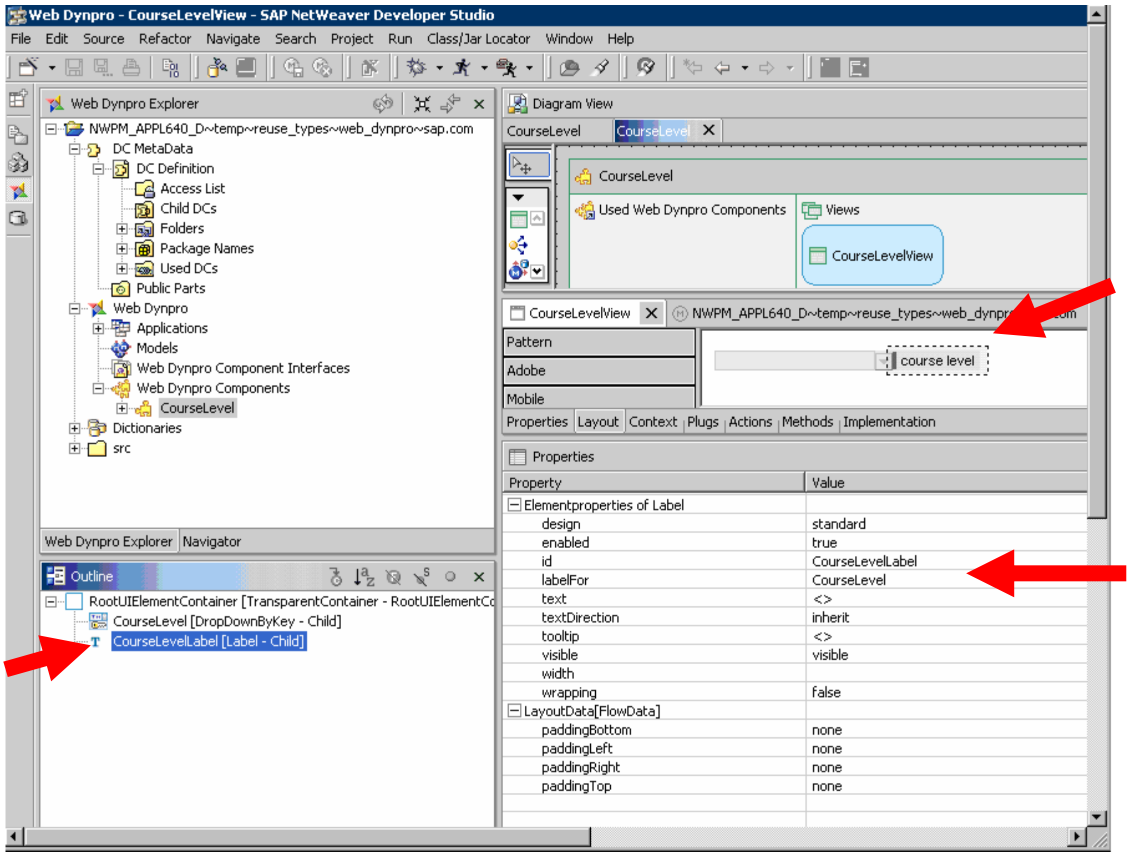
Outline:
- RootUIElementContainer [TransparentContainer - RootUIElementCo
  - CourseLevel [DropDownByKey - Child]
  - CourseLevelLabel [Label - Child]

Properties

Property	Value
Elementproperties of DropDownByKey	
enabled	true
id	CourseLevel
readOnly	false
selectedKey	
size	standard
state	normal
textDirection	inherit
tooltip	<>
visible	visible
width	
Event	
onSelect	
LayoutData[FlowData]	
paddingBottom	none
paddingLeft	none
paddingRight	none
paddingTop	none

- Field name: CourseLevel
- Field type: DropDownByKey (list box)

- There is also a second field called CourseLevelLabel. We maintain an association between the drop down list UI element CourseLevel and the label UI element CourseLevelLabel by maintain CourseLevelLabel's "labelFor" property:


Web Dynpro - CourseLevelView - SAP NetWeaver Developer Studio

File Edit Source Refactor Navigate Search Project Run Class/Jar Locator Window Help

Web Dynpro Explorer

- NWPM_APPL640_D~temp~reuse_types~web_dynpro~sap.com
  - DC MetaData
    - DC Definition
      - Access List
      - Child DCs
      - Folders
      - Package Names
      - Used DCs
    - Public Parts
  - Web Dynpro
    - Applications
    - Models
    - Web Dynpro Component Interfaces
    - Web Dynpro Components
      - CourseLevel
  - Dictionaries
  - src

Outline

- RootUIElementContainer [TransparentContainer - RootUIElementCo
  - CourseLevel [DropDownByKey - Child]
  - CourseLevelLabel [Label - Child]

Diagram View: CourseLevel — Used Web Dynpro Components; Views: CourseLevelView

CourseLevelView — NWPM_APPL640_D~temp~reuse_types~web_dynpro~sap.com

Pattern / Adobe / Mobile — course level

Properties | Layout | Context | Plugs | Actions | Methods | Implementation

Properties

Property	Value
Elementproperties of Label	
design	standard
enabled	true
id	CourseLevelLabel
labelFor	CourseLevel
text	<>
textDirection	inherit
tooltip	<>
visible	visible
width	
wrapping	false
LayoutData[FlowData]	
paddingBottom	none
paddingLeft	none
paddingRight	none
paddingTop	none

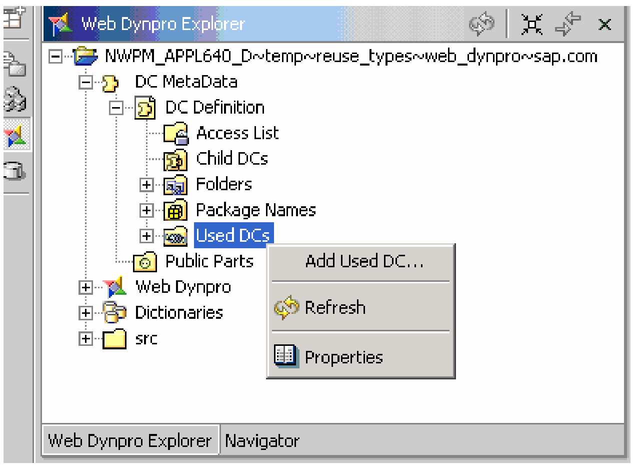
There is no data type assigned to our list box at the moment. We want to use our Dictionary Simple Type called Level to define the data type of our drop down list and label UI elements. However, before we can do this, we must…

Import Dictionary Type

If not available locally, check the Dictionary DC out from the DTR server first.

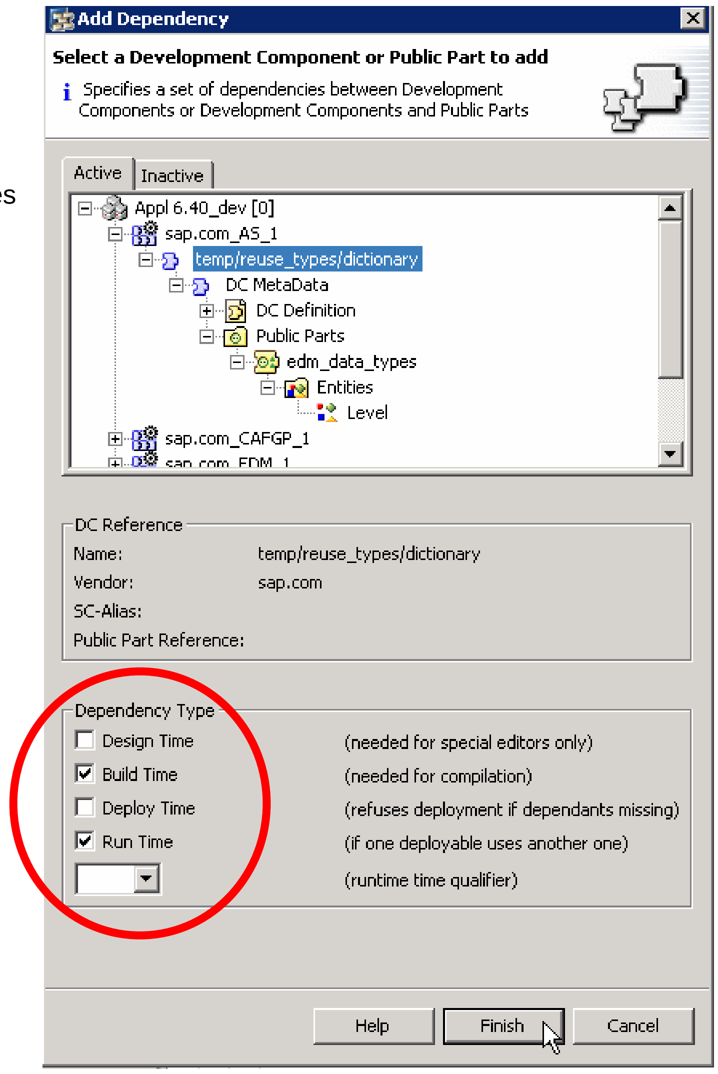
To import a simple type from another DC, in our case from a Dictionary DC, we have to establish a usage relationship between the target (Web Dynpro) DC and the source (Dictionary) DC.

Very easy: Under DC MetaData → DC Definition → Used DCs, right mouse click and select the Add Used DC option from the context menu.



In the adjacent dialog

- Navigate to and select the Dictionary DC
- Assign proper dependency types. Reusing Dictionary types requires that you select the following dependency types:
  - o Build Time (affected Web Dynpro objects need the Dictionary type for compilation)
  - o Run Time (at runtime, the Web Dynpro application dynamically links to the Dictionary data type deployed to the target server)


Add Dependency

Select a Development Component or Public Part to add

Specifies a set of dependencies between Development Components or Development Components and Public Parts

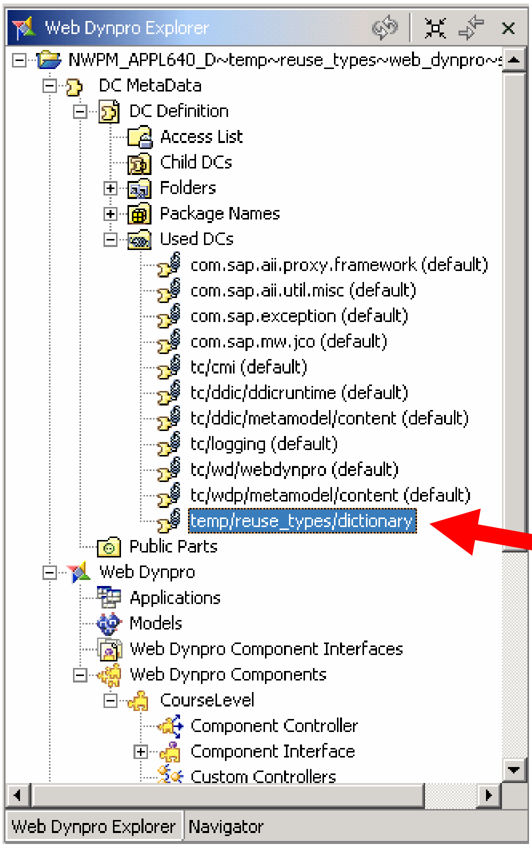
DC Reference

Name:	temp/reuse_types/dictionary
Vendor:	sap.com
SC-Alias:	
Public Part Reference:	

Dependency Type

☐ Design Time	(needed for special editors only)
☑ Build Time	(needed for compilation)
☐ Deploy Time	(refuses deployment if dependants missing)
☑ Run Time	(if one deployable uses another one)
	(runtime time qualifier)

Our source Dictionary DC now appears in the Used DCs list of our Web Dynpro DC:



```
Web Dynpro Explorer
NWPM_APPL640_D~temp~reuse_types~web_dynpro~s
  DC MetaData
    DC Definition
      Access List
      Child DCs
      Folders
      Package Names
      Used DCs
        com.sap.aii.proxy.framework (default)
        com.sap.aii.util.misc (default)
        com.sap.exception (default)
        com.sap.mw.jco (default)
        tc/cmi (default)
        tc/ddic/ddicruntime (default)
        tc/ddic/metamodel/content (default)
        tc/logging (default)
        tc/wd/webdynpro (default)
        tc/wdp/metamodel/content (default)
        temp/reuse_types/dictionary
    Public Parts
  Web Dynpro
    Applications
    Models
    Web Dynpro Component Interfaces
    Web Dynpro Components
      CourseLevel
        Component Controller
        Component Interface
        Custom Controllers
Web Dynpro Explorer | Navigator
```

As a result of this usage declaration, any objects within the public part of the Dictionary DC - in our case the simple type Level - are visible within the Web Dynpro DC.

Apply Dictionary Type as Data Type of UI Field

Finally, we are able to assign the simple type Level as the data type of the list box UI element CourseLevel.

Within a Web Dynpro view, you do not assign a data type directly to a UI element. Instead, the following procedure is required:

- First, create a context attribute in the view controller
- Second, bind context attribute’s Type property to the dictionary simple type
- Lastly, bind the UI element to the context attribute.

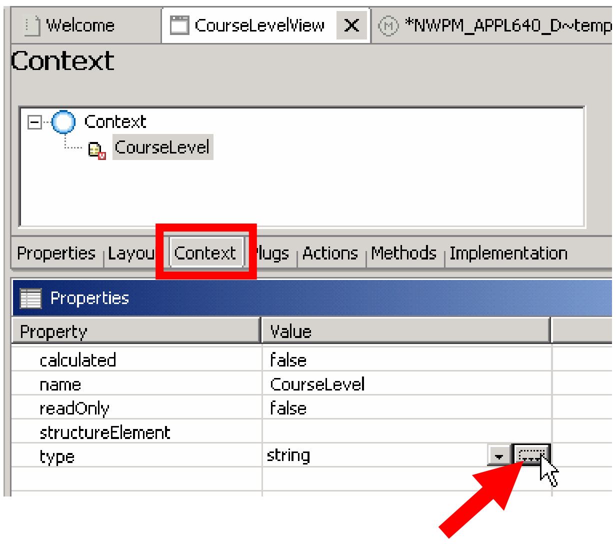
The following diagram shows this:

Web Dynpro **UI Element** ⇦⇨ Web Dynpro **Context Attribute** ⇦⇨ Dictionary **Data Type**

The UI element and the Dictionary type are already defined. So let’s look at how to create the middle item of this chain: the context attribute.

In the Context tab of the Web Dynpro view editor you can create context nodes and attributes. Below you can see a context attribute called CourseLevel. By default type property of any context attribute will default to “string”. This must be changed!

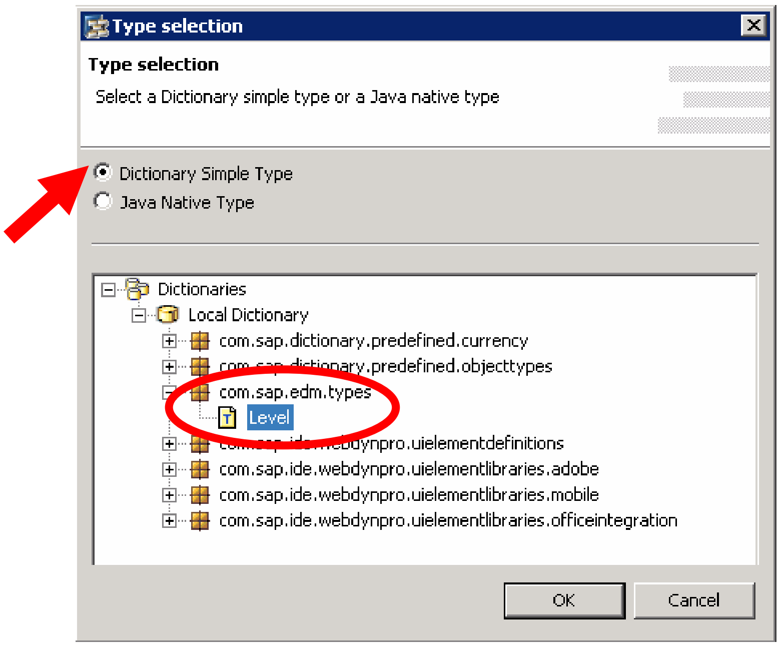
Very easy: Select the context attribute. Look for the property called type.



Property	Value
calculated	false
name	CourseLevel
readOnly	false
structureElement	
type	string

As you can see, the value of the type property is currently set to “string”. Click on the ellipsis button to the right and execute the type selection:

- Select “Dictionary Simple Type”.
- When we created the simple type, we put it in the package com.sap.edm.packages. Therefore, expand this branch and select the type Level:


Type selection

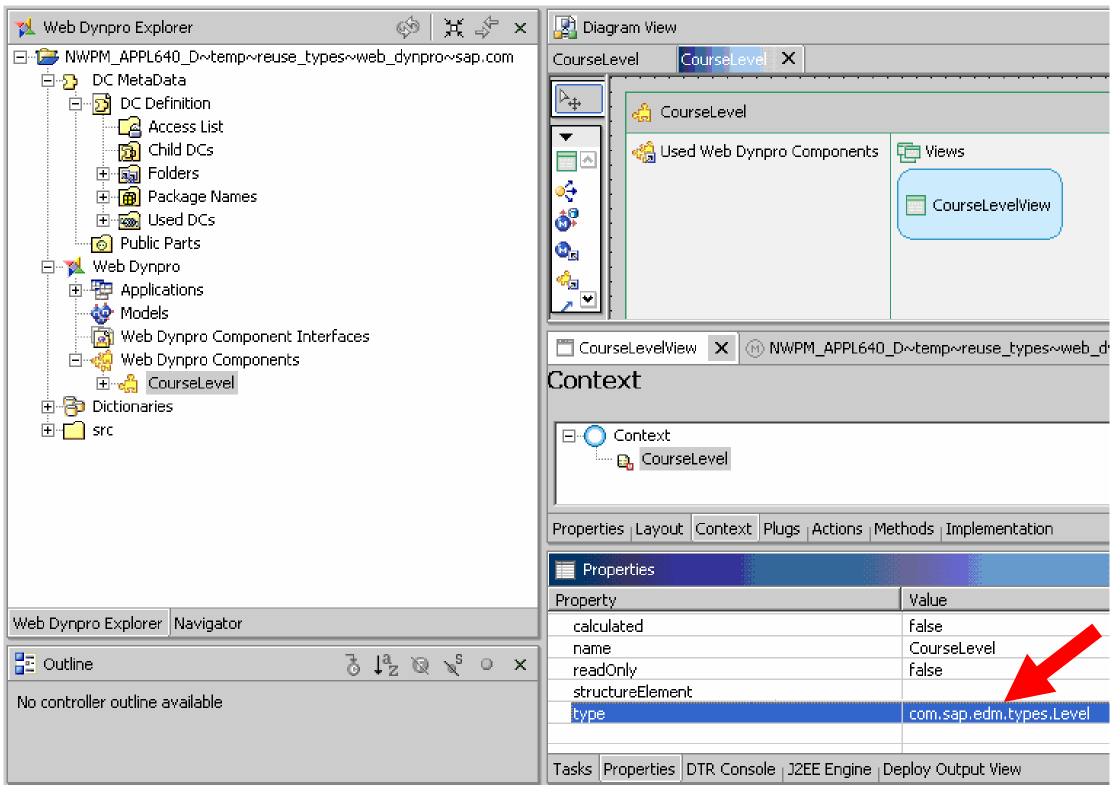
Type selection

Select a Dictionary simple type or a Java native type

(●) Dictionary Simple Type
( ) Java Native Type

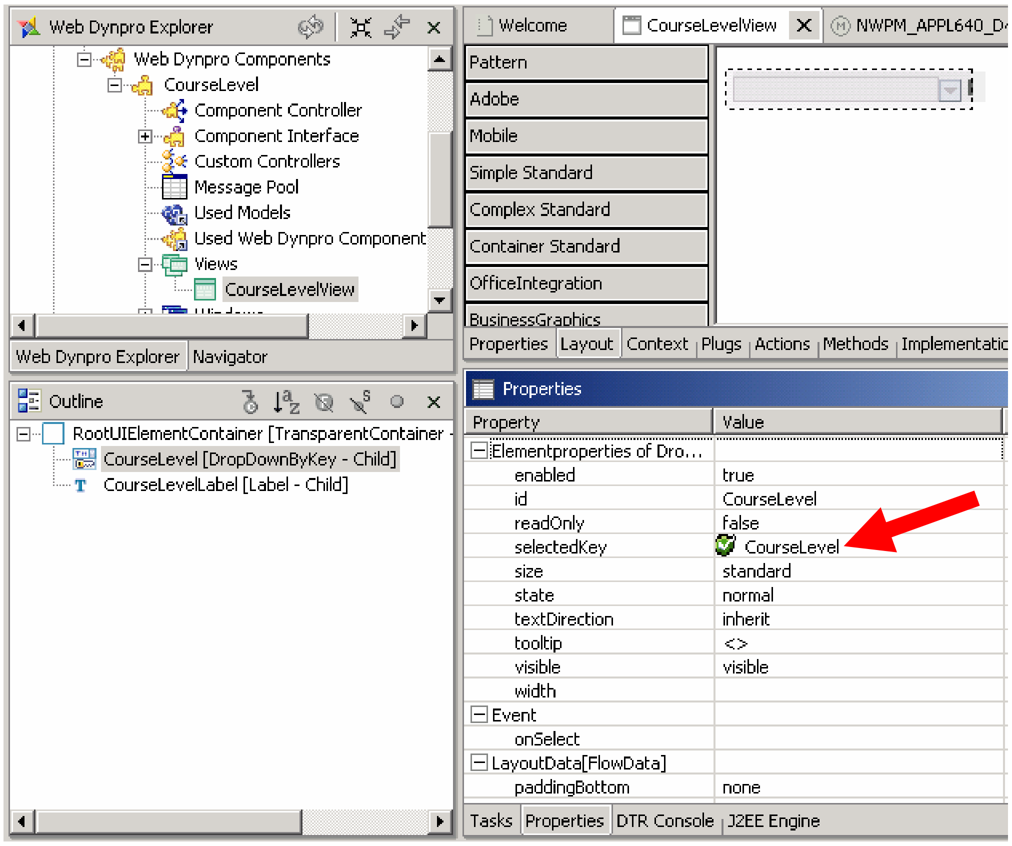
Dictionaries
Local Dictionary
com.sap.dictionary.predefined.currency
com.sap.dictionary.predefined.objecttypes
com.sap.edm.types
Level
com.sap.ide.webdynpro.uielementdefinitions
com.sap.ide.webdynpro.uielementlibraries.adobe
com.sap.ide.webdynpro.uielementlibraries.mobile
com.sap.ide.webdynpro.uielementlibraries.officeintegration
OK
Cancel

Press OK. Our Dictionary type Level is now the data type of the Web Dynpro context attribute CourseLevel:



Property	Value
calculated	false
name	CourseLevel
readOnly	false
structureElement	
type	com.sap.edm.types.Level

The remaining task is to bind the context attribute CourseLevel to the UI Element CourseLevel. As usual in Web Dynpro, we execute the binding in the layout tab of the View Editor. After this, our list box is bound to the context attribute CourseLevel:



Property	Value
Elementproperties of Dro…	
enabled	true
id	CourseLevel
readOnly	false
selectedKey	CourseLevel
size	standard
state	normal
textDirection	inherit
tooltip	<>
visible	visible
width	
Event	
onSelect	
LayoutData[FlowData]	
paddingBottom	none

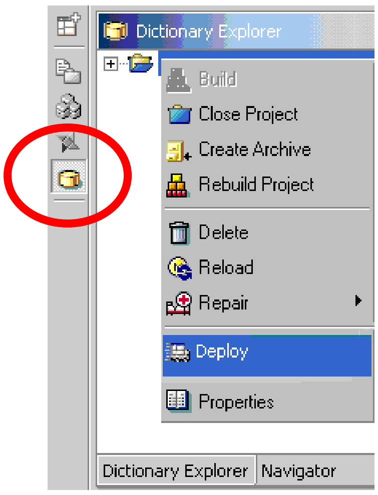
Test Web Dynpro Application

We are satisfied with our Web Dynpro UI so far. So, prior to checking it in, we test it locally.

As usual, we create a Web Dynpro application and archive and make sure the Developer Studio connects to a server instance acting as a test environment.

Deploy Dictionary Archive

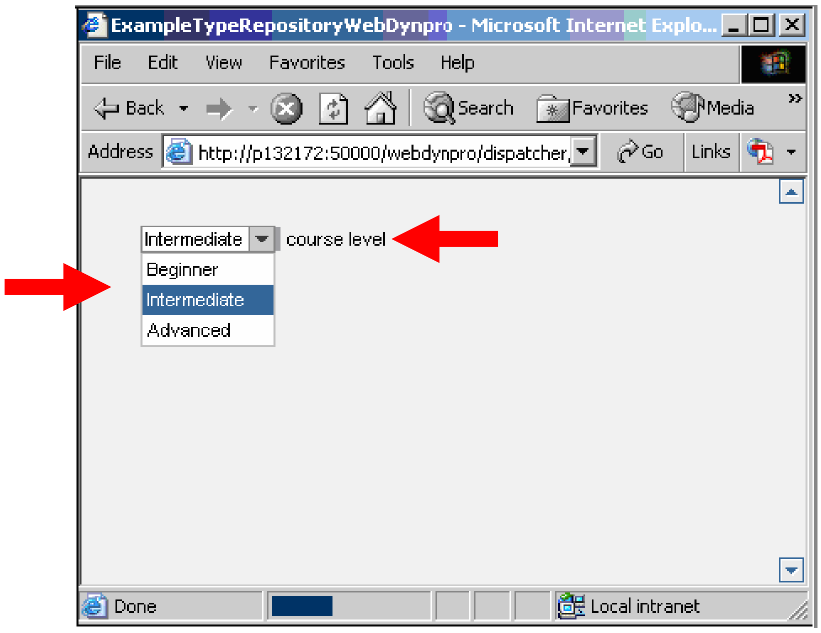
As already mentioned, the Web Dynpro framework will search for the applied Dictionary type within the server environment at runtime. To make sure the simple type Level is available on our server, we must first deploy the Dictionary archive before running the Web Dynpro application:



Deploy and Run Web Dynpro Application

Deploy and run the Web Dynpro archive.

Our type propagation is working successfully, as you can see below:



Note that:

- ✓ Enumeration values of our Dictionary type appear as the input help in our drop down list UI element.
- ✓ The label text of our Dictionary type has also been picked up by the Label UI element.

Check In Web Dynpro DC

Our Web Dynpro DC is working fine. The final steps would be to build the Web Dynpro DC and check it into the DTR server, too. This way, both DCs: The Dictionary DC and the Web Dynpro DC have been made part of the overall software transport process of the NWDI.

Conclusion

In a well designed system you should follow this principle:

Coding is bad… if you have to write the same piece of code twice.

Therefore, your aim should be to define application objects once and reuse them extensively. You learned how to apply data types defined in a Dictionary DC as types of database table columns and, on the other hand, as data types of UI fields in Web Dynpro for Java pages.

Enterprise applications usually feature a very long life cycle. As a long-term effect, applying Dictionary DCs as type repositories can simplify the maintenance of your software over years.