# Developing an International Web Dynpro Application

## Applies to:

Web Dynpro Java in SAP NetWeaver 7.0 (2004s)

## Summary

In this tutorial, the task is to develop an international Web Dynpro application that is available in English and German. As an example, we will use a car rental application that displays different texts, depending on the booking. You will learn to use *Simple Types* and *Message Pool.*

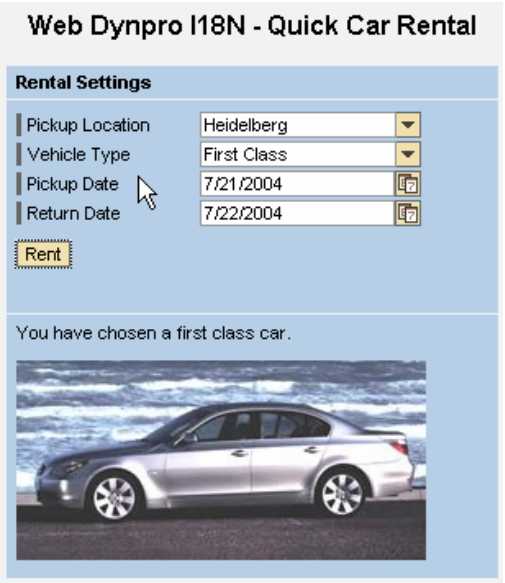
**Author(s):** SAP NetWeaver Product Management

**Company:** SAP AG

**Created on:** 01 February 2008

## Table of Contents







## Sample Project

In the remainder of this tutorial, you will gradually add to this initial project template until is a complete Web Dynpro project, which is also available for separate download.

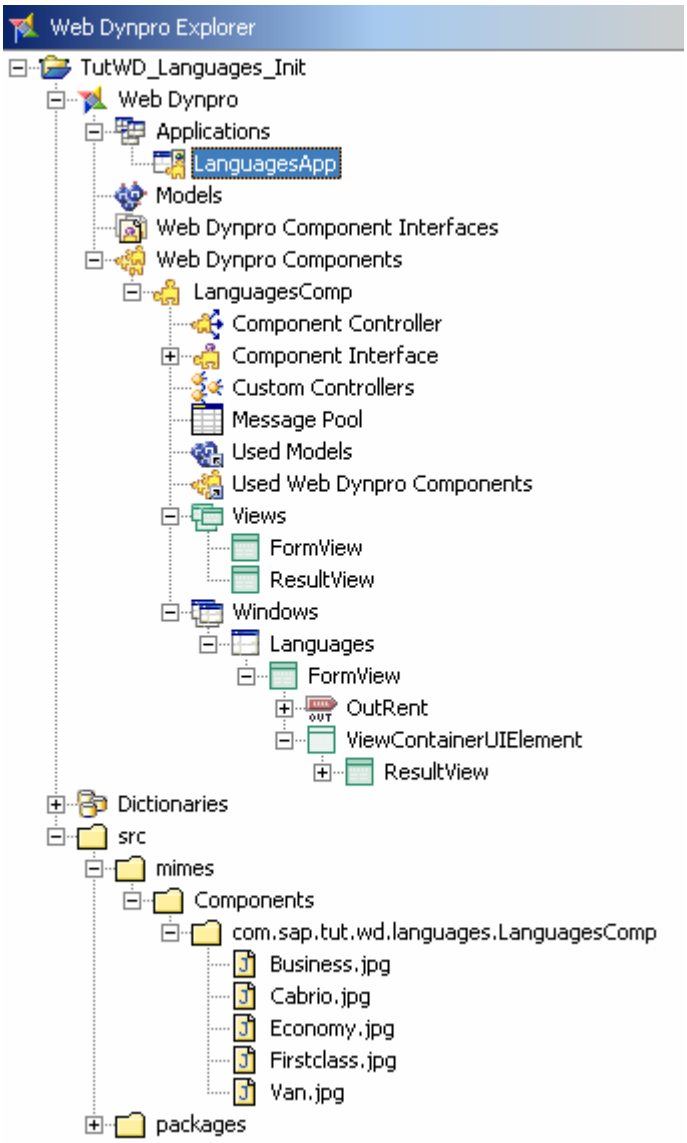
- *TutWD_Languages_Init.zip*: Initial Web Dynpro project template
- *TutWD_Languages.zip:* Complete Web Dynpro project

### Importing a Project Template

If you are already familiar with Web Dynpro and the SAP NetWeaver Developer Studio you probably know how to import a project, otherwise you can find the necessary steps described at help.sap.com: Importing a Project

### Initial Project Structure

Once the Web Dynpro project template *WDJ_Languages_Init* has been imported, the following project structure should be displayed in the Web Dynpro Explorer:




### Creating Texts for the Original Language

A central aspect of internationalization is the translatability of the texts that are displayed on the user interface. For this reason, an international application must not contain any language-specific text elements in its source code. You must define these texts in the Message Editor.

To make the translation process smoother, we recommend that you store any frequently used language-specific texts that you have defined in the Layout editor, such as labels or table headers, in *Simple Types*.

In Web Dynpro, the text elements that you create at design time are isolated and grouped automatically in `*.xlf` files.

Web Dynpro is not currently linked to an SAP translation system that can extract the automatically generated `*.xlf` files and import them into an existing SAP-based translation system. For this reason, the following procedure includes a workaround for the internationalization and translation of the application.

### Developing the Application - Steps

In the following tutorial, you add all texts to the application that you want to display on the user interface. In the next step, you internationalize these texts.

Structure of the Tutorial:

- Create texts for the original language
  a. Create the simple types
  b. Create the context and action
  c. Complete the layout of the application
  d. Create the warning message and implement *onActionRent()*
  e. Create dynamic texts and complete the application
- Translate text resources into other languages
- Define language-specific application properties

### Creating Simple Types

To display selection options in the dropdown list for the vehicle type, add the simple type **VehicleType** to the application. Among other things, simple types are suitable for encapsulating frequently used texts that are defined in the Layout Editor at design time.

1. In the *TutWD_Languages_Init* project, open the **Data Types** node (choose *TutWD_Languages_Init* → *Dictionaries* → *Local Dictionary* → *Data Types*).
2. Open the context menu for the **Simple Types** node and choose *Create Simple Type*.
3. Under *Simple Type Name*, enter **VehicleType**. Under *Package*, enter **com.sap.tut.wd.languages.simpletypes** and choose *Finish* to confirm.

   The editor for the new simple type *VehicleType* opens automatically.
4. Choose the *Enumeration* tab page.
5. To add a new enumeration, choose *New...*
6. Under *Enumeration Value*, enter **E**. Under *Enumeration Text*, enter **Economy**. Choose *Finish* to confirm.
7. Repeat steps 5 and 6 to create more enumerations as shown in the table below.

| Value | Description |
|---|---|
| B | Business Class |
| F | First Class |
| C | Convertible |
| V | Van |

8. Switch to the *Representation* tab page.
9. Under *Field Label*, enter **Vehicle Type**.
10. Under *Quick Info:*, enter **Choose a vehicle type**.

    To display selection options in the dropdown list for the pickup location, add the simple type **Location** to the application.

11. Repeat steps 2-10 to create the simple type *Location* in the package **com.sap.tut.wd.languages.simpletypes**. Use the following properties:

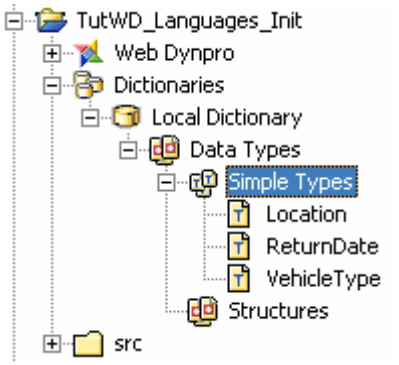
| *Enumeration* | |
|---|---|
| **Value** | **Description** |
| HD | Heidelberg |
| F | Frankfurt |
| B | Berlin |
| M | Muenchen |

| *Representation* | |
|---|---|
| Field Label | Pickup Location |
| Quick Info | Choose a pickup location |

To make it clearer how you encapsulate frequently used interface element texts in a simple type, create the simple type *ReturnDate*.

12. Create the simple type **ReturnDate** in the package **com.sap.tut.wd.languages.simpletypes**.
13. Switch to the *Definition* tab page and choose *date* in the dropdown list for the build-in. Choose *OK*.
14. Switch to the *Representation* tab page. Under *Field Label*, enter **Return Date**. Under *Quick Info*, enter **Choose a date .**

You have created the three simple types *Location, ReturnDate*, and *VehicleType*:




## Creating the Context and Action

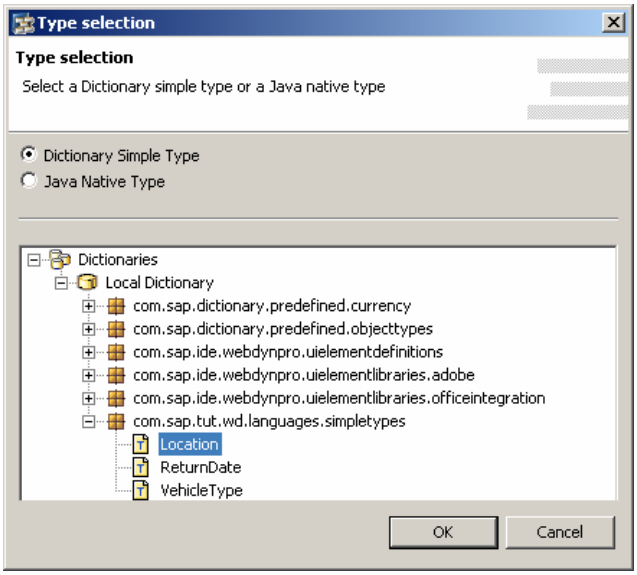
### Creating the Context

To bind the dropdown lists for the vehicle type and pickup location, and the input fields for the pickup date and return date, to the corresponding context attributes, you must first create these attributes.

1. Open the **FormView**. (choose *TutWD_Languages_Init* → *Web Dynpro* → *Web Dynpro Components* → *LanguagesComp* → *Views* → *FormView*).
2. Switch to the *Context* tab page.
3. Create the value attribute *Location*.

   The *Location* context attribute is given the data type of the new simple type *Location* as its value.

4. Select the *Location* context attribute. On the *Properties Page*, choose  to open the dialog for selecting the simple type for the *type* property.
5. Select *Dictionary Simple Type* and choose *Dictionaries* → *Local Dictionary* → *com.sap.tut.wd.languages.simpletypes*. Choose the *Location* simple type and confirm by choosing *OK*.




6. Create further value attributes:

| Context | Type | Property | Value |
|---|---|---|---|
| PickupDate | Value | *type* | date |
| ReturnDate | Value attribute | *type* | com.sap.languages.simpletypes.ReturnDate<br><br>**Note:** We recommend that you create context attributes that contain a simple type with text resources if the text resources of the simple type appear often on the interface. |
| VehicleType | Value | *type* | com.sap.languages.simpletypes.VehicleType |

## Creating an Action

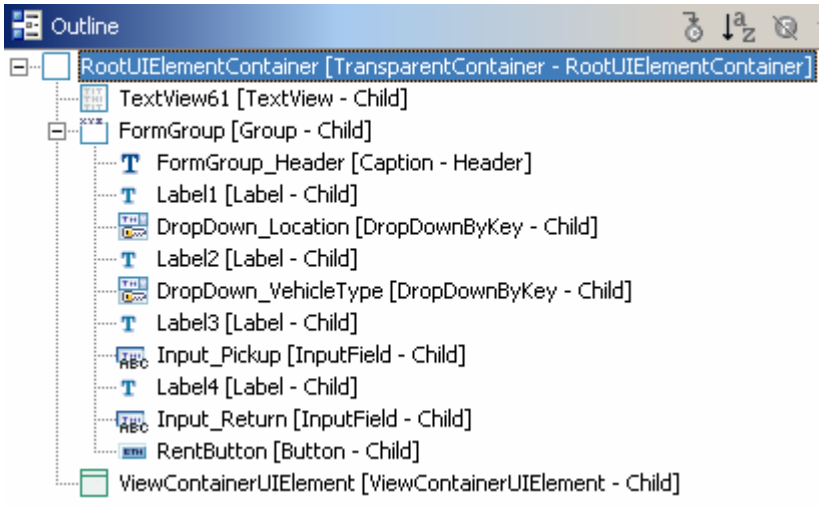
To confirm the booking and display *ResultView*, create the **Rent** action.

1. In *FormView*, switch to the *Actions* tab page.
2. Create an action by choosing *New*. Give this action the name **Rent** and the text **Rent**. Choose *Finish*.

## Completing the Layout of the Application

In the following section, you complete the interface elements.

1. In **FormView**, switch to the *Layout* tab page. The following interface elements already exist:



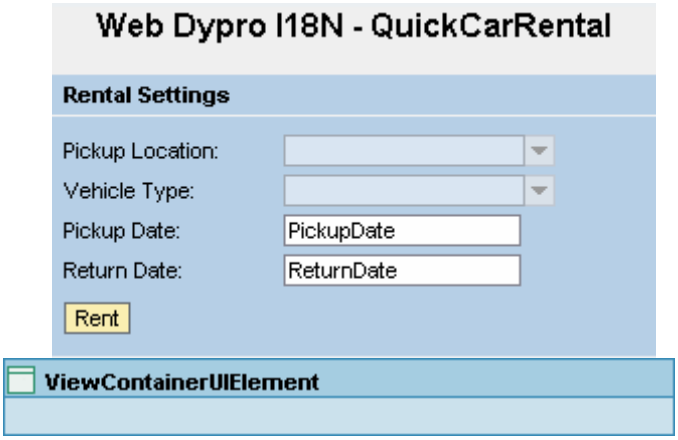

2. Add new properties for the following interface elements or bind them to the appropriate context elements and actions:

| Properties | Value |
|---|---|
| **TextView61** of the type *TextView* | |
| *text* | Web Dynpro I18N – Quick Car Rental |
| **FormGroup_Header** with *Caption* type | |
| *text* | Rental Settings |
| **DropDown_Location** with *DropDownByKey* type | |
| *selectedKey* | Location |
| **Label1** of the type *Label* | |
| *labelFor* | *DropDown_Location*<br><br>**Note:** Since *DropDown_Location* is bound to the *Location* context attribute (with the *Location* simple type created previously), the field text from the simple type is used as the default text. |
| **DropDown_VehicleType** of the type *DropDownByKey* | |
| *selectedKey* | VehicleType |
| **Label2** of the type *Label* | |
| *labelFor* | DropDown_VehicleType |
| **Input_Pickup** with *InputField* type | |
| value | PickupDate |

| | |
|---|---|
| **Label3** of the type *Label* | |
| *labelFor* | Input_Pickup |
| *text* | Pickup Date<br>**Note:** Since Input_Pickup is bound with a context attribute without a simple type, you must define the text here. |
| **Input_Return** with *InputField* type | |
| *value* | ReturnDate |
| **Label4** with *Label* type | |
| *labelFor* | Input_Return |
| **RentButton** of the type *Button* | |
| *onAction* | Rent |
| *text* | **Note:** The empty default value remains. Since the button is bound to the Rent action and the Rent text is assigned to this action, this text is used for this button automatically. |

3. Save the current status of the metadata.

You have completed the layout of *FormView*:




## Creating a Warning Message and Implementing onActionRent()

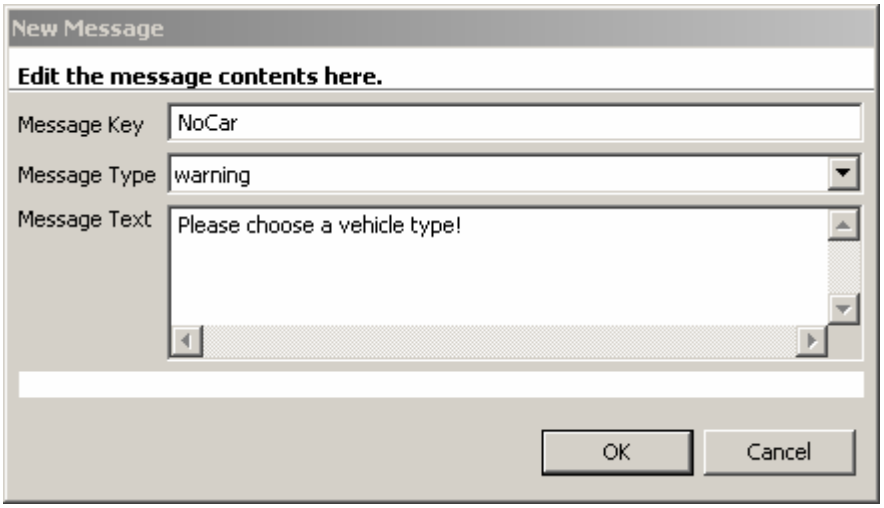
If the user of the example application chooses *Rent* without first specifying a vehicle type, an appropriate warning message must appear. If the user has selected a vehicle type, *ResultView* displays the vehicle type.

In the following procedure, you implement a warning message in the message pool and the *onActionRent()* method. You can display the warning message on the user interface using the *IWDMessageManager* interface. You can access this interface in the controller code using the shortcut variable *wdComponentAPI*.

Messages of the type: ERROR, WARNING, or STANDARD are stored as constants in the interface *IMessage<ComponentName>.java* which is generated by the generation framework on component level.

### Creating a Warning Message

1. Open the message pool of the *LanguagesComp*. (LanguagesComp → Message Pool)
2. Add a new message by choosing *Open Message Editor* from the context menu and select the first icon  in the the Message Editor.
3. Under *Message Key*, enter **NoCar**. Under *Message Type*, enter **warning**. Under *Message Text*, enter **Please choose a vehicle type**, and choose *OK*.




4. Save the current status of the metadata.

### Implementation of the Method onActionRent()

1. Open *FormView* and switch to the *Implementation* tab page.
2. Add the following source code to the method *onActionRent()*:

**onActionRent()**

```
//@@begin onActionRent(ServerEvent)
String vehicleType = wdContext.currentContextElement().getVehicleType();
//if no vehicleType was choosen
if (vehicleType == null){
    IWDMessageManager msg = wdComponentAPI.getMessageManager();
    msg.reportMessage(IMessageLanguagesComp.NO_CAR, null, false);
 else{   //if a vehicleType was selected navigate to ResultView
    wdThis.wdFirePlugOutRent(vehicleType);
}
```

## Creating dynamic texts and complete the application

### Defining Dynamic Language-Specific Texts in the Message Pool

1. Open the message pool for *LanguagesComp*.
2. Add new texts to the message pool by choosing  for each new text.

   The table shows you the properties of each new text:

| Message Key | Message Type | Message Text |
|---|---|---|
| text_E | text | You have chosen an economy class car. |
| text_F | text | You have chosen a first class car. |
| text_B | text | You have chosen a business class car. |
| text_C | text | You have chosen a cabriolet. |
| text_V | text | You have chosen a van. |

### Extracting New Texts from the Message Pool in the Program Code

You can now use the *onPlugInResult()* method to extract the new texts from the message pool.

1. Open **ResultView** and switch to the *Implementation* tab page.
2. Add the following source code to the method *onPlugInResult()* and that followed choose **Source > Organize Imports** from the context menu:

**onPlugInResult()**

```
//@@begin onPlugInResult(ServerEvent)
String text, image;

    //provides access to translatable texts from Message Pool
    IWDTextAccessor textAccessor = wdComponentAPI.getTextAccessor();

    if (vehicleType.equals("E")) {
      //text = "You have chosen an economy class car.";
      text = textAccessor.getText("text_E");
      image = "Economy.jpg";
    }
    else if (vehicleType.equals("F")) {
      //text = "You have chosen a first class car.";
      text = textAccessor.getText("text_F");
      image = "Firstclass.jpg";
    }
    else if (vehicleType.equals("B")) {
      //text = "You have chosen a business class car ";
      text = textAccessor.getText("text_B");
      image = "Business.jpg";
    }
    else if (vehicleType.equals("C")) {
      //text = "You have chosen a cabriolet.";
      text = textAccessor.getText("text_C");
      image = "Cabrio.jpg";
    }
    else { //Van
      //text = "You have chosen a van.";
      text = textAccessor.getText("text_V");
      image = "Van.jpg";
    }
    wdContext.currentContextElement().setText(text);
    wdContext.currentContextElement().setImage(image);  //@@end
}
```

> **Note:** The pictures are included in the *TutWD_Languages_Init* project. You can access them in the Package Explorer by choosing *TutWD_Languages_Init* → *src* → *mimes* → *Components* → *com.sap.tut.wd.languages.LanguagesComp*.

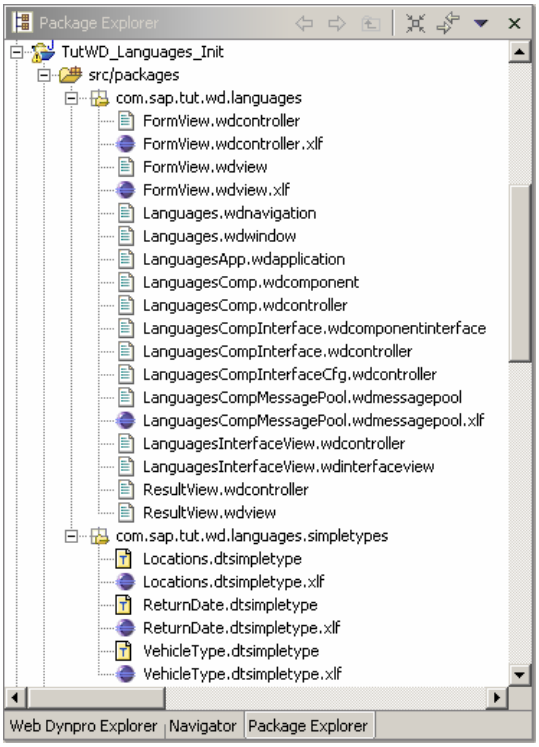
3. Save the current status of the metadata. At this point the application is already runnable.

## Translating Text Resources into Other Languages

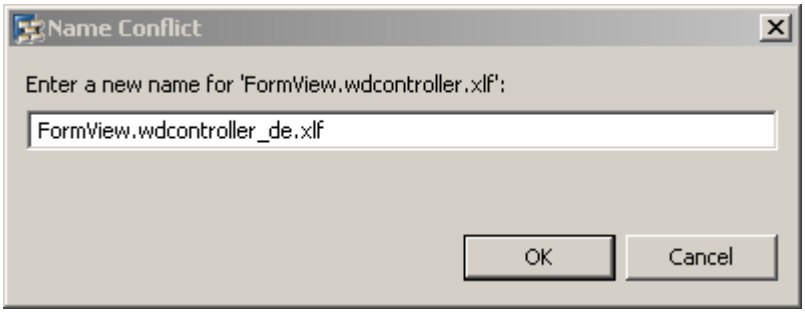
In Web Dynpro, the text elements created at design time are isolated in resource bundles (.xlf files). The following section describes how you add English and German texts to the example application. You can also add other languages (such as Spanish) to the application. Just use the appropriate translations and language keys in the following steps.

### Copying the *.xlf Files

1. To copy the automatically generated *.xlf files, and save them under new language keys, switch to the Package Explorer.
2. Open the *src/packages* directory (choose *TutWD_Languages_Init* → *src/packages*). Open both *com.sap.tut.wd.languages* and *com.sap.tut.wd.languages.simpletypes*. These two directories contain the *.xlf files that belong to the application.



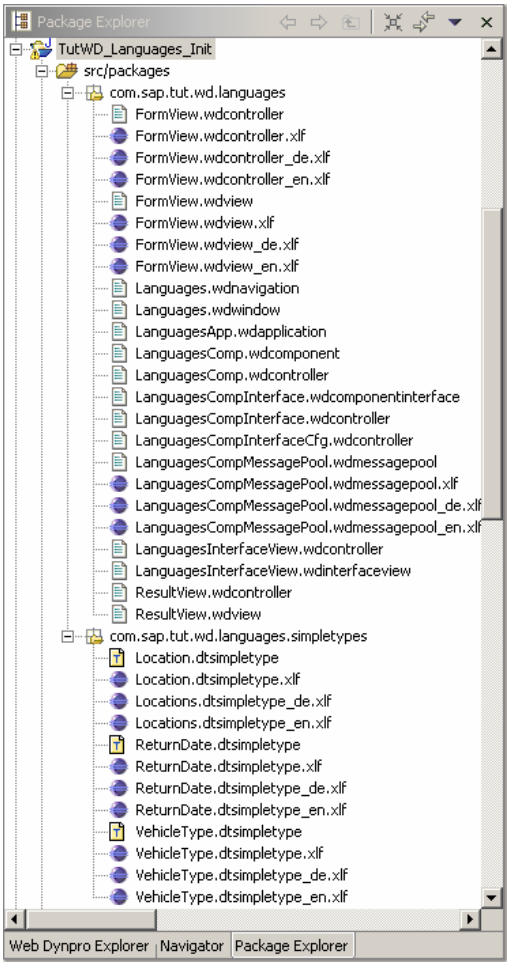

3. In the context menu of the *FormView.wdcontroller.xlf* file, choose *Copy*.
4. In the context menu of the *com.sap.tut.wd.languages* directory, choose *Paste*.
5. In the next dialog box, add **_de** before **.xlf** and choose *OK*.




6. Repeat steps 5-6 for English (enter _**en** before **.xlf**).
7. Repeat steps 3-6 for the following files:

   Insert the copies for *Location*, *VehicleType*, and *ReturnDate* in the directory *com.sap.tut.wd.languages.simpletypes*.

   - FormView.wdview.xlf
   - LanguagesCompMessagePool.wdmessagepool.xlf
   - Location.dtsimpletype.xlf
   - VehicleType.dtsimpletype.xlf
   - ReturnDate.dtsimpletype.xlf

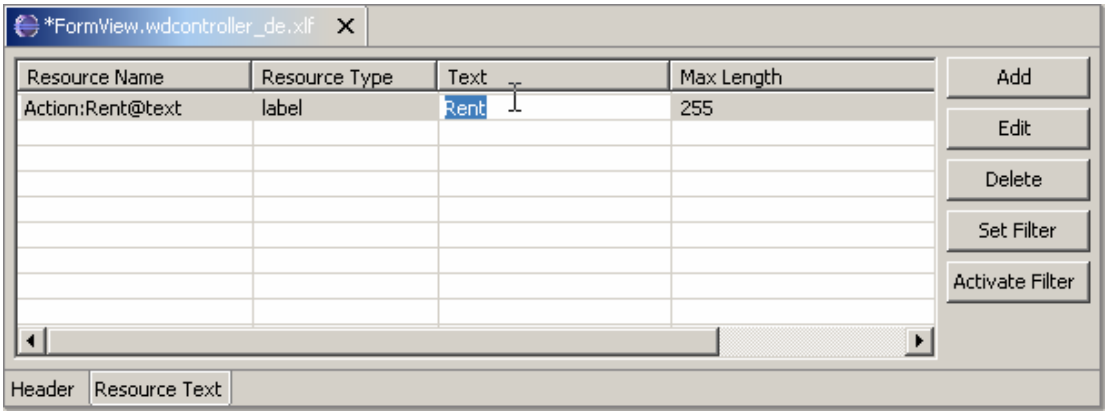


## Translating the *_de.xlf Files

The *TutWD_Languages_Init* project has been created in the project language American English. You define this default language when you create a new Web Dynpro project. For this reason, all text input in the tutorial has been in English up to now. The *.xlf files that are not assigned to a specific language are used as default values.

Since the *.xlf files without language keys are already defined for English texts, you do not need to translate the resource bundles with the language key _**en**.

1. Open the file *FormView.wdcontroller_de.xlf* by double-clicking it. This file is displayed in the S2X Editor.
2. On the *Header* tab page, change the source language to German by selecting this language from the dropdown list.
3. Switch to the *Resource Text* tab page. In the *Action:Rent@text* row, go to the *Action:Rent@text* column and translate *Rent* as **Mieten**. Confirm with **Enter**.




4. Save the change by choosing the icon in the toolbar.
5. Repeat steps 1-4 for the following files:
   a. FormView.wdview_de.xlf
   b. LanguagesCompMessagePool.wdmessagepool_de.xlf
   c. Location.dtsimpletype_de.xlf
   d. VehicleType.dtsimpletype_de.xlf
   e. ReturnDate.dtsimpletype_de.xlf

   Use the following translations:

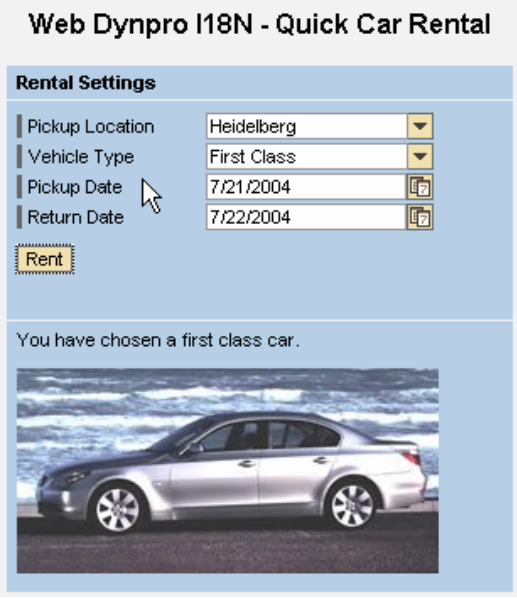
| English Text | German Text |
|---|---|
| **FormView.wdview_de.xlf** | |
| *Web Dynpro I18N - Quick Car Rental* | Web Dynpro I18N – Mietwagenbuchung |
| *Pickup Date* | Abholdatum |
| *Rental Settings* | Mietwagenbuchung |
| **LanguagesCompMessagePool.wdmessagepool_de.xlf** | |
| *You have chosen a cabriolet.* | Sie haben ein Cabriolet gemietet. |
| *You have chosen a van.* | Sie haben einen Van gemietet. |
| *Please choose a vehicle type!* | Bitte geben Sie einen Fahrzeugtyp an! |
| *You have chosen a first class car.* | Sie haben einen Luxusklasse-Wagen gemietet. |

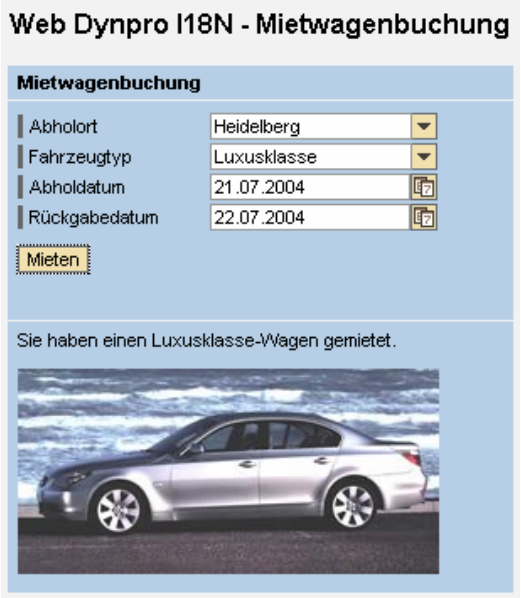
| | |
|---|---|
| *You have chosen an economy class* | Sie haben einen Economyklasse-Wagen gemietet. |
| *You have chosen a business class car.* | Sie haben einen Mittelklasse-Wagen gemietet. |
| **VehicleType.dtsimpletype_de.xlf** | |
| *Cabriolet* | Cabriolet |
| *Economy* | Economy |
| *Van* | Van |
| *Business* | Mittelklasse |
| *First Class* | Luxusklasse |
| *Vehicle Type* | Fahrzeugtyp |
| *Choose a vehicle type.* | Wählen Sie einen Fahrzeugtyp. |
| **ReturnDate.dtsimpletype_de.xlf** | |
| *ReturnDate* | Rückgabedatum |
| *Choose a date* | Wählen Sie ein Rückgabedatum. |
| **Location.dtsimpletype_de.xlf** | |
| *Pickup Location* | Abholort |
| *Choose a pickup location* | Wählen Sie einen Abholort. |

You have made the example application available in English and German.

**Note:** However, the new and modified *.xlf files only take effect when you execute ***Reload*** for the Web Dynpro project. To see the effect of internationalization you can change your browsers language.

In the Internet Explorer browser, you can set the language by choosing *Tools → Internet Options...* On the *General* tab page, choose *Languages...* You can now prioritise, add and delete languages.



## Related Content

How to Define Text Mappings for a Web Dynpro Java Application

Simple Types and Enumerations

Programming User Messages

Internationalization of Web Dynpro Projects

Configuring a Web Dynpro Application

## Copyright