Developing International Web Dynpro Applications

SAP NetWeaver 04

Copyright

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Developing International Web Dynpro Applications 5
Importing a Project Template 6
Creating Texts for the Original Language 8
Creating the Simple Types 9
Creating the Context and Action 11
Completing the Layout of the Application 13
Creating a Warning Message and Implementing onActionRent() 14
Create dynamic texts and complete the application 16
Translating Text Resources into Other Languages 18
Defining language-specific application properties 22
Executing and Testing the Application 24

Developing International Web Dynpro Applications

The Task

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 various different texts, depending on the booking.

The basic framework of the application is provided by the project template *TutWD_Languages_Init*. You will add language-specific resources to the application and learn about the most important aspects of developing international Web Dynpro applications.

Objectives

By the end of this tutorial, you will be able to:

- ✓ internationalize Web Dynpro applications

- ✓ create simple types that contain the interface texts
- ✓ save and read language-specific texts in the message pool.

Prerequisites

Systems, Installed Applications, and Authorizations

- SAP NetWeaver Developer Studio is installed on your PC.
- You have access to the SAP J2EE Engine.

Knowledge

- Basic knowledge of the Java programming language
- Knowledge of programming of Web Dynpro applications

Next step:

Importing a Project Template [page 6]

Importing a Project Template

On the SAP Developer Network (SDN) at http://sdn.sap.com, the following Web Dynpro projects are available for this tutorial:

- The project template *TutWD_Languages_Init* (the starting point for this tutorial)
- The completed Web Dynpro project *TutWD_Languages* (corresponds to the project *TutWD_Popup_Init* after completion of the tutorial)

The projects are available for download in corresponding zip files under the category *Home* → *Developer Areas* → *Web Application Server* → *Web Dynpro*.

Prerequisites

- You have access to the SAP Developer Network (http://sdn.sap.com).
- You have installed the SAP NetWeaver Developer Studio.

Procedure

Importing the Project Template to the SAP NetWeaver Developer Studio

1. Call the SAP NetWeaver Developer Network by using the URL http://sdn.sap.com and log on with the corresponding user ID and password. If you do not have a user ID, you must register before you can proceed.
2. Download the zip file `TutWD_Languages_Init.zip` containing the project template *TutWD_Languages_Init* and save the zip file to any directory on your local hard disk or directly in the work area of the SAP NetWeaver Developer Studio.
3. Extract the content of the zip file `TutWD_Languages_Init.zip` in the work area of the SAP NetWeaver Developer Studio or in any directory on your local hard disk.
4. Open the SAP NetWeaver Developer Studio.
5. Import the Web Dynpro project *TutWD_Languages_Init:*
   a. In the menu, choose *File* → *Import*.

b. In the next window, choose *Existing Project into Workspace* and choose *Next* to confirm.

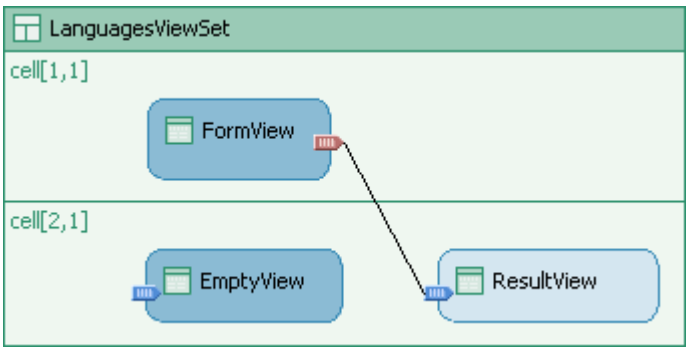
c. Choose *Browse*, open the folder in which you saved the project *TutWD_Languages_Init*, and select the project.

d. Choose *Finish* to confirm.

The Web Dynpro project *TutWD_Languages_Init* appears in the Web Dynpro Explorer for further processing and completion of the tutorial.

Tabular Project Structure

Once you have imported the project template to the SAP NetWeaver Developer Studio, you will see the following structure in the Web Dynpro Explorer.

Web Dynpro Project Structure
Web Dynpro Project: **TutWD_Languages_Init**
Web Dynpro application: **LanguagesApp**
Web Dynpro component: **LanguagesComp**
View: **FormView** *FormView* contains the basic framework for the car rental form.
View: **ResultView** *ResultView* contains the basic framework for displaying the rental car.
Window: **Languages** *FormView* and *ResultView* are embedded in the *Languages* window. Since you do not want to display *ResultView* until a booking has been confirmed, an *EmptyView* is embedded in the window. 
src → *mimes* → *Components* → *com.sap.tut.wd.languages.LanguagesComp* This directory contains the pictures for the *QuickCarRental* application. (`Business.jpg`; `Cabrio.jpg`; `Economy.jpg`; `Firstclass.jpg`; `Van.jpg`)

The initial Web Dynpro project *TutWD_Languages_Init* cannot yet run in the Web browser, since the language-specific resources are still missing.

Next step:

Creating Texts for the Original Language [page 8]

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*.

This tutorial is concerned with the following language-specific texts:

- Language-specific resources that you define **at design time** in the NetWeaver Developer Studio, such as the following
    - Labels for interface elements
    - Error messages
- Language-specific resources that you use **dynamically** in the source code

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 R/3-based translation system. For this reason, the following procedure includes a workaround for the internationalization and translation of the application.

Process Flow

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. The procedure *Defining Language-Specific Application Properties* describes how the language display of the *QuickCarRental* application is determined.

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

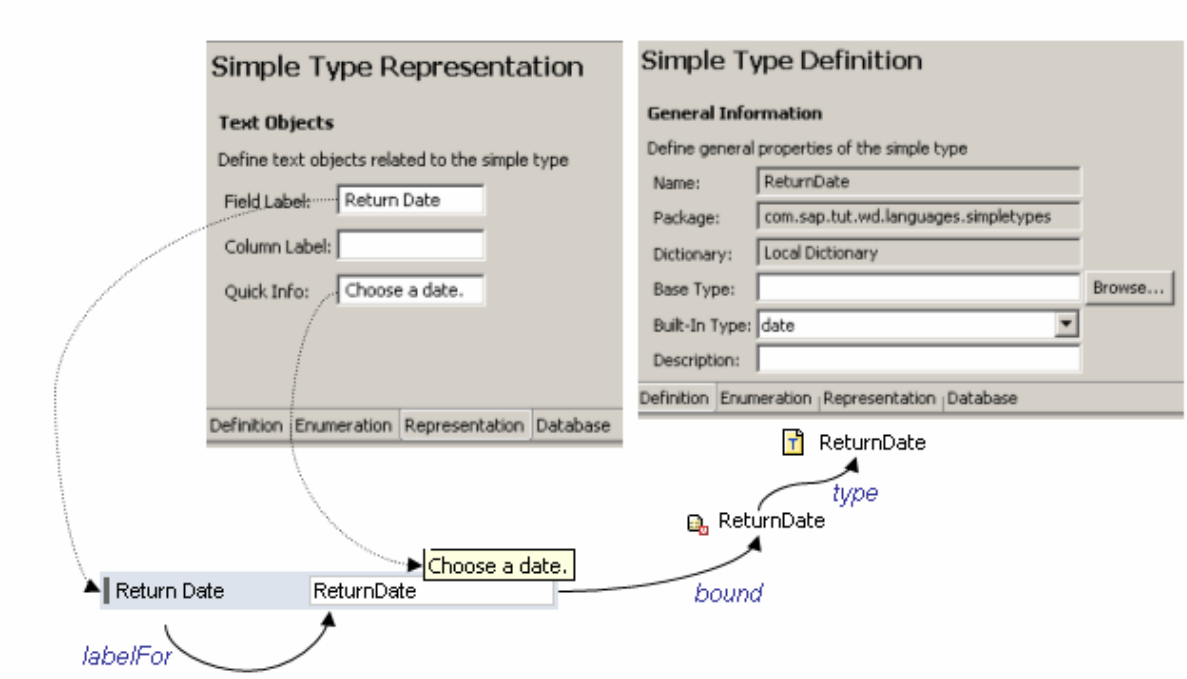
Next step:

Creating the Simple Types [page 9]

Creating the Simple Types

Among other things, simple types are suitable for encapsulating frequently used texts that are defined in the Layout Editor at design time.

Figure 1 shows you an example of how to use a simple type. The `ReturnDate` simple type has the type **date** and encapsulates the texts *Return Date* (a label text) and *Choose a date*. (a tooltip text). You can now assign various context attributes to this simple type, such as `ReturnDate`. You can also bind an input field to this context attribute. In this way, the tooltip text of the simple type is used for the input field. If you define a label for the input field (you must set the *labelFor* property for the label), the label text of the simple type is used.

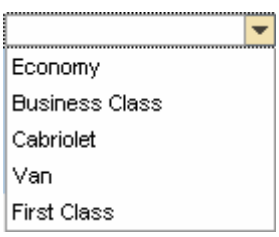


You can also define enumerations for simple types. You can also assign this simple type to a context attribute. For example, you can bind this context attribute to a dropdown list. In this way, all values included in the simple type can be displayed in the dropdown list.

Simple Type Enumeration

Value	Description
E	Economy
B	Business Class
C	Cabriolet
V	Van
F	First Class

Dropdown List



Procedure

Creating the Simple Types

To display selection options in the dropdown list for the vehicle type, add the simple type **VehicleType** to the application.

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`**. 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.

Value	Description
B	Business Class
F	First Class
C	Cabriolet
V	Van

8. Switch to the *Representation* tab page.
9. Under *Field Label*, enter **`Vehicle Type`**.

   Later, this value is used as the default value for the corresponding label. (*Column Label* defines the default value for a table header.)
10. Under *Quick Info:*, enter **`Choose a vehicle type`**.

    This text is used as the default value for the tooltip.

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:

Enumaration	
Value	Description
HD	Heidelberg
F	Frankfurt
B	Berlin
M	Munich

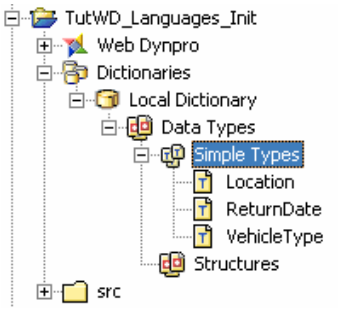
Representation	
Field Label	Pickup Location
Quick Info	Choose a pickup location.

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.`**

Result

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



Next step:

Creating the Context and Action [page 11]

Creating the Context and Action

Procedure

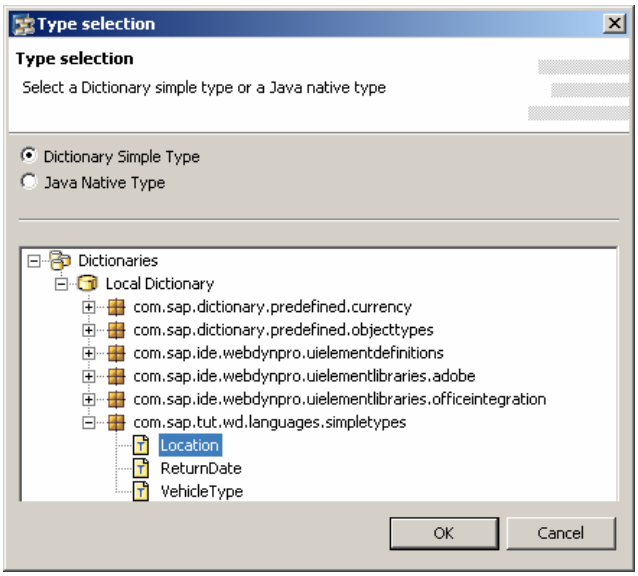
Creating the Context

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 Element	Type	Property	Value
PickupDate	Value attribute	*type*	date
ReturnDate	Value attribute	*type*	`com.sap.languages.simpletypes.ReturnDate`  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 attribute	*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*.

Next step:

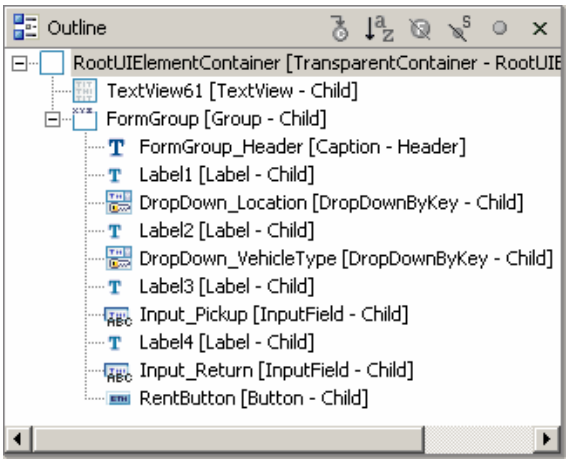
Completing the Layout of the Application [page 13]

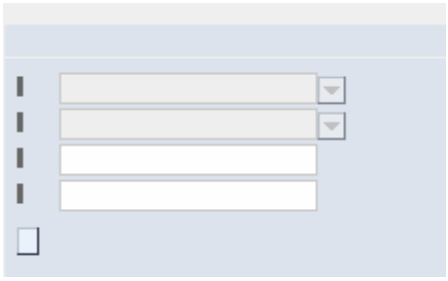
Completing the Layout of the Application

In the following section, you complete the interface elements.

Procedure

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:

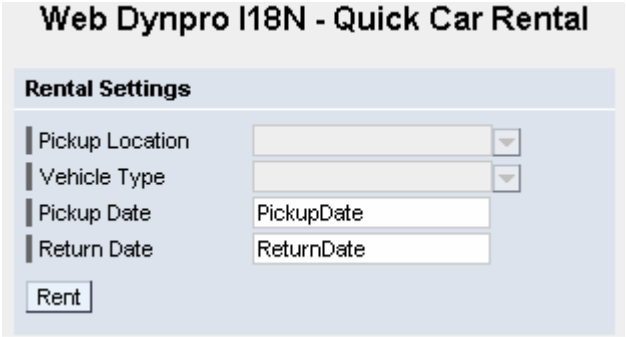
Properties	Value
TextView61 of the type *TextView*	
text	Web Dynpro I18N – Quick Car Rental
FormGroup_Header with *Caption* type	
text	Rental Settings
DropDown_Location with *DropDownByKey* type	
selectedKey	Location
Label1 of the type *Label*	
labelFor	*DropDown_Location* Since *DropDown_Location* is 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 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	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.

Result

You have completed the layout of *FormView*:



Next Step:

Creating the Warning Message and Implementing onActionRent() [page 14]

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

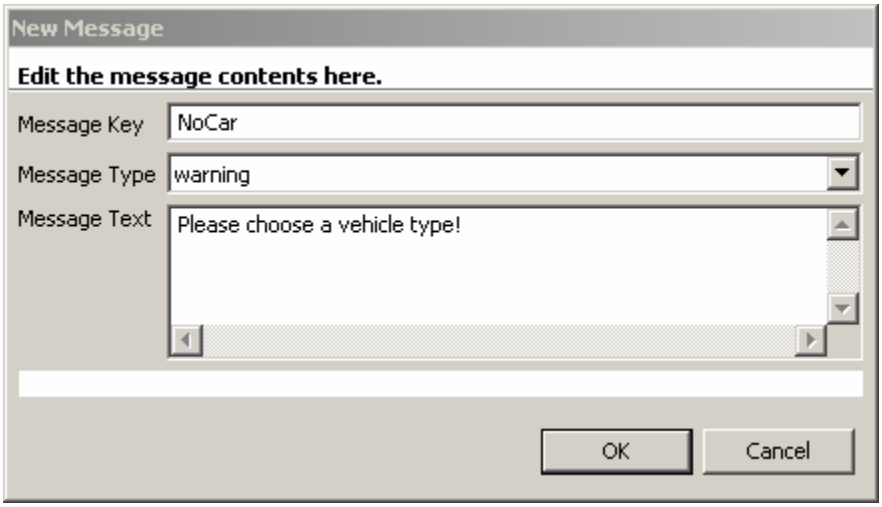
the **`IWDMessageManager`** interface. You can access this interface in the controller code using the shortcut variable `wdComponentAPI`.

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

Procedure

Creating a Warning Message

1. Open the message pool of the *LanguagesComp*. (LanguagesComp → Message Pool)
2. Add a new message by choosing  .
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()

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

onActionRent()

```
public void onActionRent(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent ) {
  //@@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);
  }
  //@@end
}
```

You no longer need to make the imports required to display the message, since they have already been defined in the project template.

Next Step:

Creating Dynamic Texts and Completing the Application [page 16]

Create dynamic texts and complete the application

ResultView displays a picture and text that correspond to the vehicle type chosen by the user. You realize this dynamic screen change in the `onPlugRentIn()` method.

Do not define any dynamic interface texts for international applications in the source code. It is extremely difficult to extract these texts and make them non-language-specific.

Web Dynpro gives you the option of saving dynamic texts in the message pool. The text elements in the message pool are saved automatically to an xlf file. This isolates the texts in groups, which makes the translation process easier.

You also have the (more difficult) option of extracting the implemented language-specific texts from the source code or saving them directly in a *.properties file. For more information, see Externalizing Strings [extern] or Internationalization Service [extern].

In the following procedure, you save these language-specific texts in the message pool. In the controller code, you can access the texts of the message pool using the **`IWDTextAccessor`**.

The **`IWDTextAccessor`** enables you to access the texts of messages of the type *IWDMessage* (*standard*, *error* or *warning*) and the message type *text*:

- //Returns a localized text for the given key
  java.lang.String getText(java.lang.String key)
- //Returns a formatted text whose localized text is determined by the
  //given key, parameterized by the given parameters
  java.lang.String getText(java.lang.String key, java.lang.Object[] parameters)
- //Returns a localized text for the given message
  java.lang.String getText(IWDMessage message)
- //Returns a formatted text whose localized text is determined by the
  //given message, parameterized by the given parameters
  java.lang.String getText(IWDMessage message, java.lang.Object[] parameters)

Procedure

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()`:

onPlugInResult()

```
public void
onPlugInResult(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent, java.lang.String vehicleType ) {
  //@@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
}
```

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.

Result

You have created a complete example application from the *TutWD_Languages_Init* project. You can now start and run the application.

Next step:

Translating Text Resources into Other Languages [page 18]

Translating Text Resources into Other Languages

In Web Dynpro, the text elements created at design time are isolated in resource bundles.

*.xlf files are created for the following Web Dynpro elements. The corresponding texts are saved and isolated in these files:

MessagePool:	<Web Dynpro Component Name>MessagePool.wdmessagepool.xlf
View:	<View Name>.wdview.xlf <View Name>.wdcontroller.xlf (text of the actions, for example)
window	<Window Name>.wdwindow.xlf (title of the window, for example)
SimpleType:	<SimpleType Name>.dtsimpletype.xlf

To internationalize the Web Dynpro application, copy the automatically generated *.xlf files and save them under a new name in the same directory. The new name must meet the following convention:

- <old name>_<language key>.xlf

Choose an appropriate language key. For example, if you are creating *.xlf files for German, use the language key **de**.

You can edit and translate these new *.xlf files in the S2X Editor.

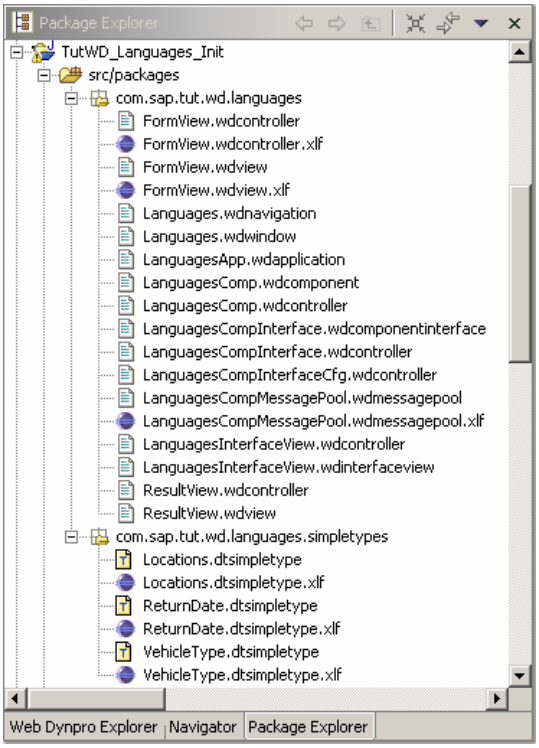
Procedure

The following section describes how you add English and German texts to the example application.

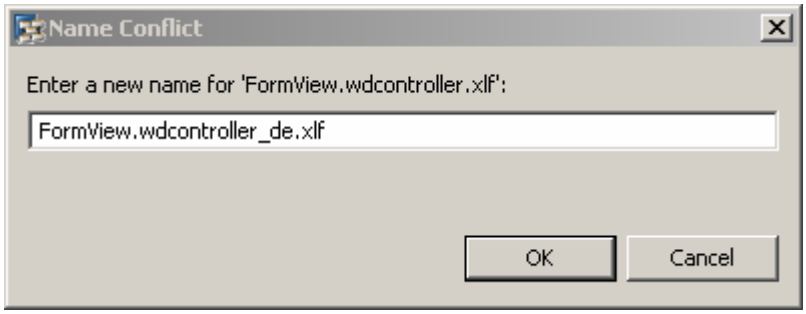
You can also add other languages (such as Spanish) to the application. Just use the appropriate translations and language keys in the following steps.

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.

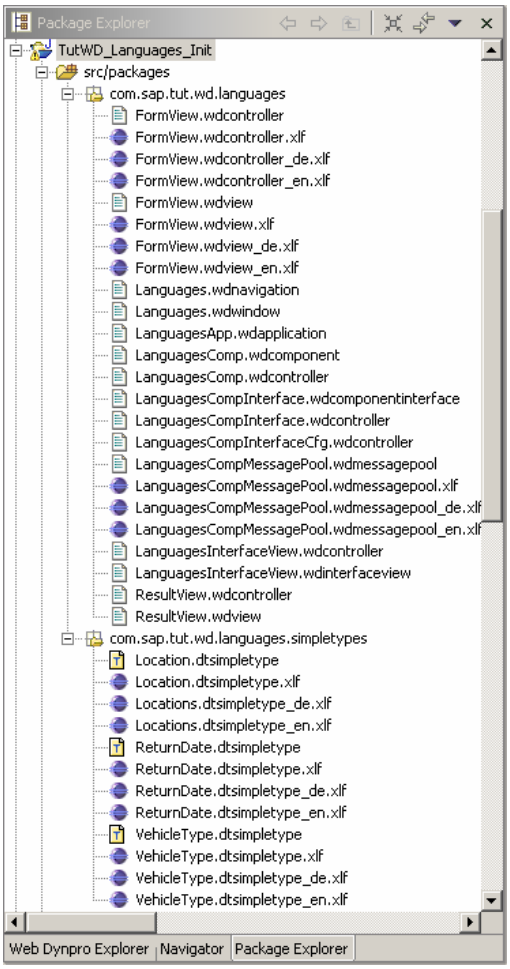


3. In the context menu of the *FormView.wdcontroller.xlf* file, choose *Copy*.
4. In the context menu of the *com.sap.tut.wd.languages* directory, choose *Paste*.
5. In the next dialog box, add **_de** before `.xlf` and choose *OK*.



6. Repeat steps 5-6 for English (enter **_en** before `.xlf`).
7. Repeat steps 3-6 for the following files:
   - `FormView.wdview.xlf`
   - `LanguagesCompMessagePool.wdmessagepool.xlf`
   - `Location.dtsimpletype.xlf`
   - `VehicleType.dtsimpletype.xlf`
   - `ReturnDate.dtsimpletype.xlf`

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

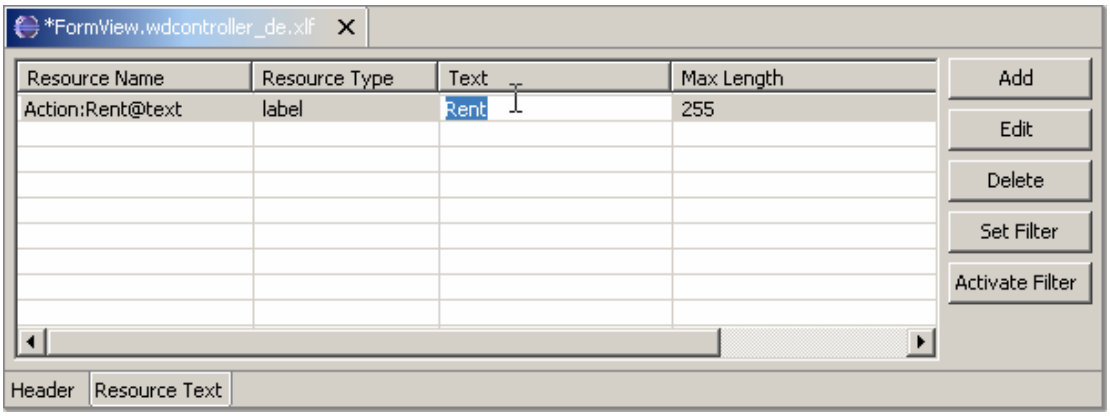


Translating the *_de.xlf Files

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.`
You have chosen a first class car.	`Sie haben einen Luxusklasse-Wagen gemietet.`
You have chosen an economy class car.	`Sie haben einen Economyklasse-Wagen gemietet.`
You have chosen a business class car.	`Sie haben einen Mittelklasse-Wagen gemietet.`
Please choose a vehicle type!	`Bitte geben Sie einen Fahrzeugtyp an!`
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.`

Result

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

However, the new and modified *.xlf files only take effect when you execute *Reload* for the Web Dynpro project. The procedure is described under Executing and Testing the Application [page 24].

Next Step:

Defining Language-Specific Application Properties [page 22]

Defining language-specific application properties

In the Web Dynpro application, you can define the properties of the application under *Application Properties*. These properties can include the sequence in which the language is displayed. Web Dynpro provides you with the following predefined application properties for internationalization:

- **Authentication**

  If the user has been authenticated, the application is displayed in the language specified in the user profile.
- **DefaultLocale**

If a user does not need authentication to use the application, this setting specifies the language that is displayed for anonymous users if no language preference is defined in the Web browser. You can use the search process described below to determine the session locale used by the Web Dynpro runtime environment when starting an application.

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 add and delete languages.

If the Web Dynpro application requires user authentication, the language indicator is determined through the data of the user who is logged on. If the Web Dynpro application does not require user authentication, the language key of the used browser is used for the selection of the properties file. If the language indicator is not set, the language preference `sap-`

`locale` defined for the application is used, then the one for the system, and finally the one of the current Java Virtual Machine (JVM).

Example

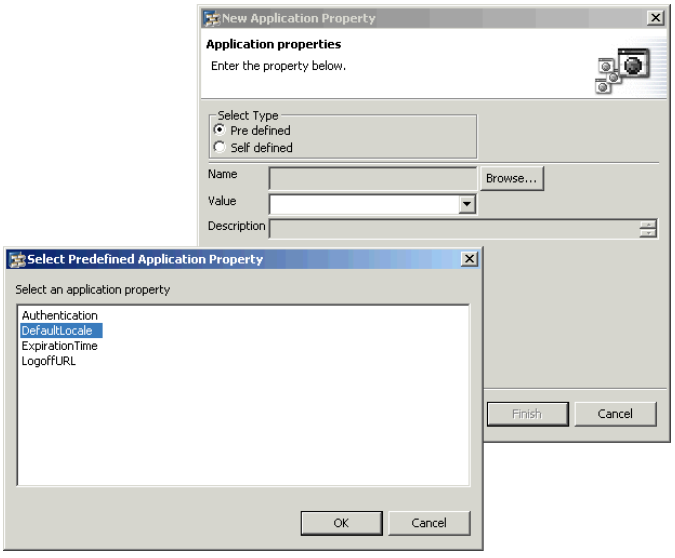
User	Language indicator of the user	Language preference of the browser	Language preference of the application	Language preference of the system	Language prefere of the JVM
Authenticated	de	en	fr	it	ru
Anonymous	-	en	fr	it	ru
Anonymous	-	-	fr	it	ru
Anonymous	-	-	-	it	ru
Anonymous	-	-	-	-	ru

For more information, refer to Search Process for Determining the Required Resource Bundle [extern].

Procedure

To define *DefaultLocale* of a Web Dynpo application, proceed as follows:

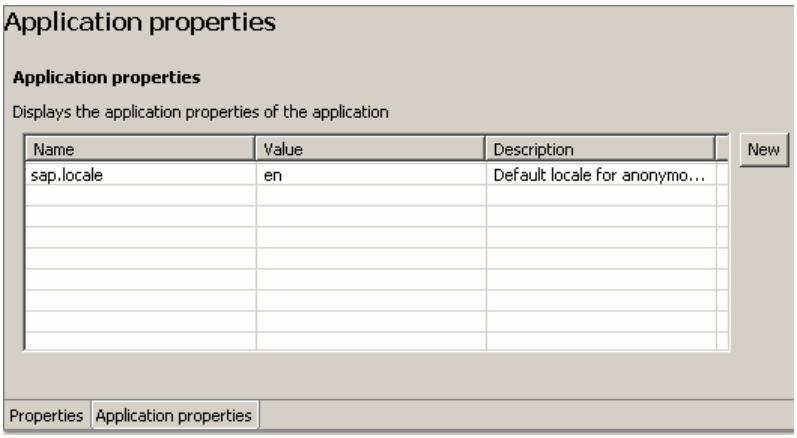
1. Open the Web Dynpro application *LanguagesApp* (choose *TutWD_Languages_Init* → *WebDynpro* → *Applications*).
2. Choose *New*. In the next dialog box, choose *Browse...*
3. In the next dialog box, select *DefaultLocale*, then choose *OK*.



4. Under *Value*, enter **en**. Confirm by choosing *Finish*. This defines English as the default language.

Result

You have defined the default locale for the Web Dynpro application.



Next step:

Executing the Internationalized Application TutWD_Languages_Init [page 24]

Executing and Testing the Application

Now that you have reached this stage, you can start the fully developed example application in the Web Browser as described below.

Prerequisites

☐ You have made sure that the SAP J2EE Engine has been launched.

Procedure

1. Save the current state of the metadata for your project by choosing *Save All Metadata*.
2. In the *Web Dynpro Explorer*, open the context menu for the project node **`TutWD_Language_Init`** and choose *Reload*.

   The newly added xlf files are only included in the deployment if you explicitly execute *Reload*.
3. Open the context menu and choose *Rebuild Project*.
4. In the Web Dynpro Explorer, in the context menu of the application object **`LanguagesApp`**, choose *Deploy New Archive and Run*.

Making Language Settings in the Web Browser

5. In the Web browser, add the languages English and German.

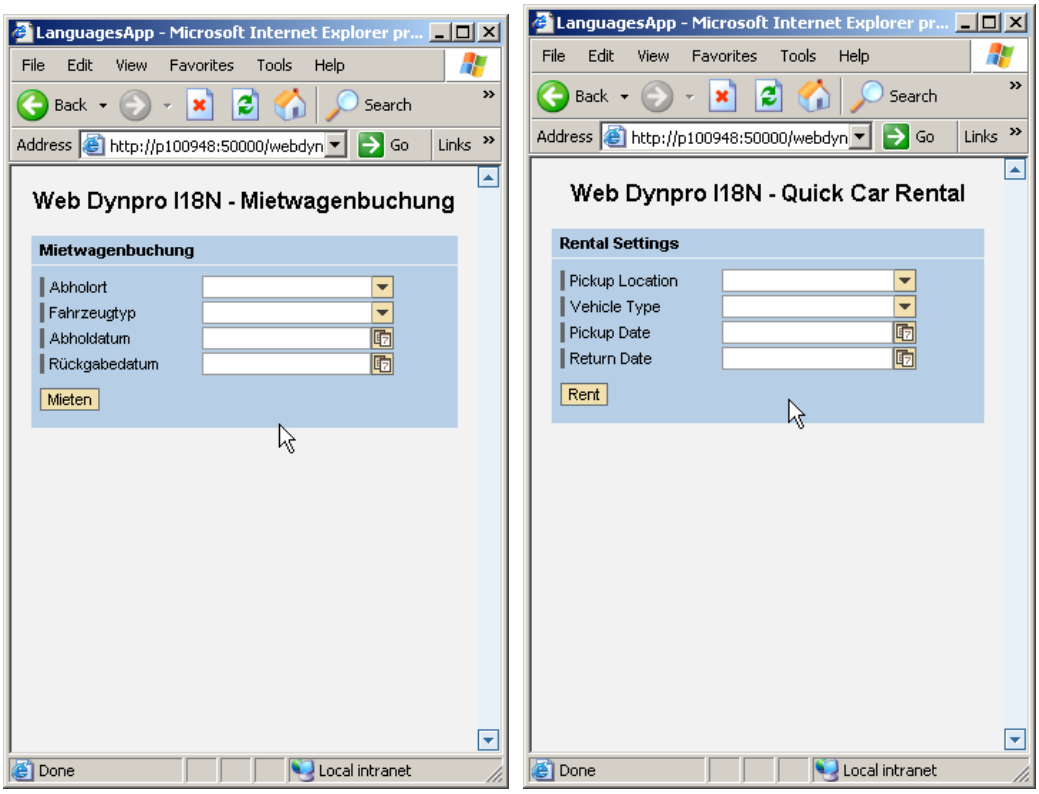
In the Internet Explorer browser, you can set the language by choosing *Tools* → *Internet Options...* On the *General* tab page, choose *Languages...* You can now add and delete languages.

6. Start the application again.

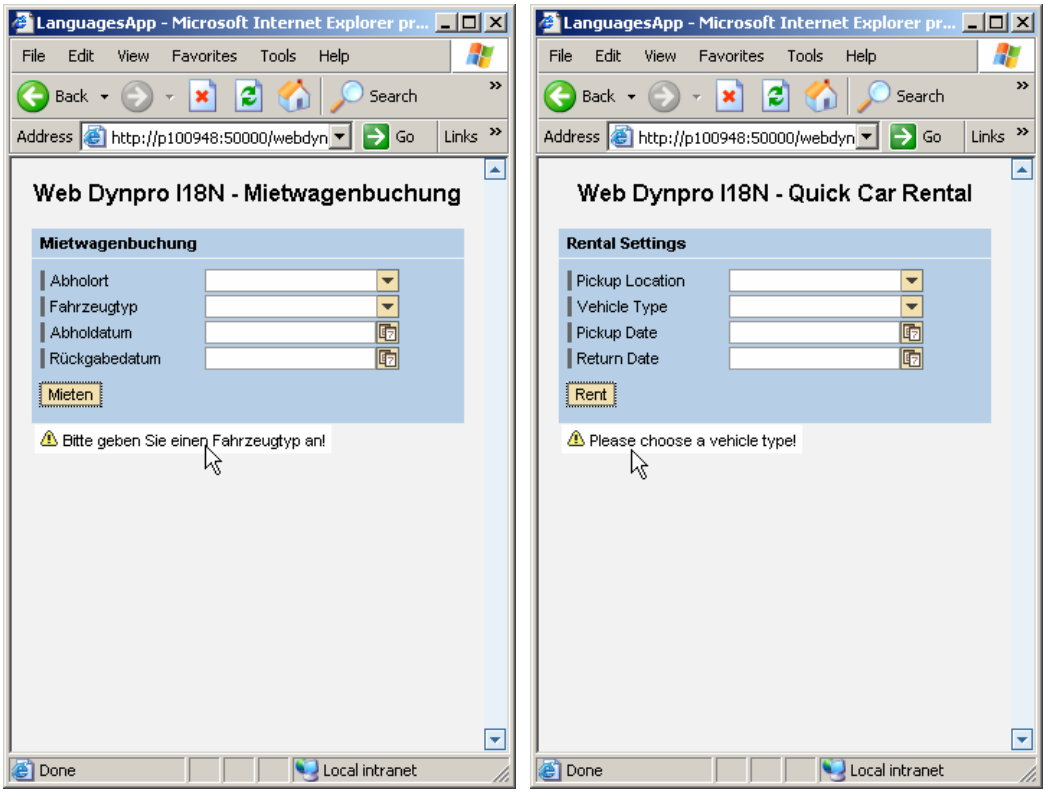
Result

The English or German variant of the example application is displayed in the Web browser, depending on your language setting and the priority of the language.

The *QuickCarRental* application opens in the browser window:

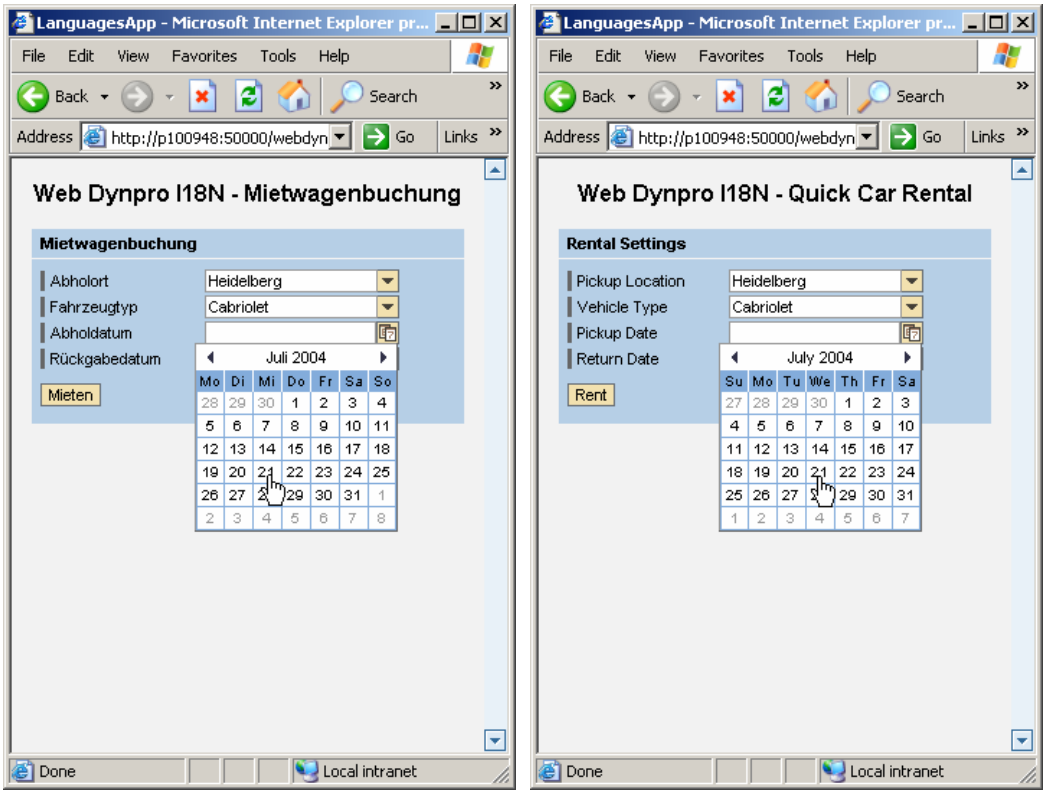


If you choose *Rent* (English) or *Mieten* (German) without selecting a vehicle type, an appropriate error message appears at the bottom of the window:

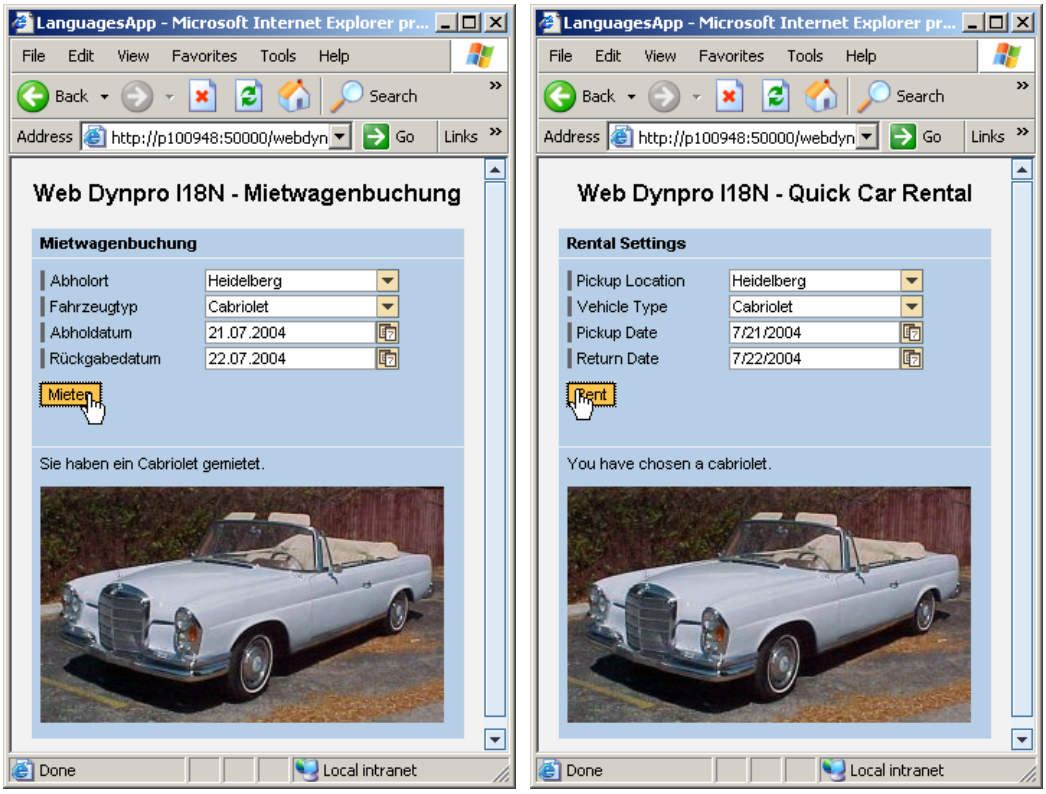


Rent a vehicle.

Use the calendar. It is also language-specific.



After you confirm your booking, the rented vehicle is displayed.


LanguagesApp - Microsoft Internet Explorer pr...
File Edit View Favorites Tools Help
Back Search
Address http://p100948:50000/webdyn Go Links
Web Dynpro I18N - Mietwagenbuchung
Mietwagenbuchung
Abholort Heidelberg
Fahrzeugtyp Cabriolet
Abholdatum 21.07.2004
Rückgabedatum 22.07.2004
Mieten
Sie haben ein Cabriolet gemietet.
Done Local intranet
LanguagesApp - Microsoft Internet Explorer pr...
File Edit View Favorites Tools Help
Back Search
Address http://p100948:50000/webdyn Go Links
Web Dynpro I18N - Quick Car Rental
Rental Settings
Pickup Location Heidelberg
Vehicle Type Cabriolet
Pickup Date 7/21/2004
Return Date 7/22/2004
Rent
You have chosen a cabriolet.
Done Local intranet