# Common-Controls Localization

Version 1.5 - Last changed: 30. Januar 2005

Common Controls

**Published by:**
SCC Informationssysteme GmbH
64367 Mühltal

Tel: +49 (0) 6151 / 13 6 31 0
Internet www.scc-gmbh.com

Product Site
http://www.common-controls.com

# Table of content

# 1 Introduction

This document treats the possibilities for creating multi-lingual applications with the Common-Controls Framework.

The following table gives an initial overview of the relevant control and form elements:

| Control | Language-dependent components |
|---|---|
| ListControl | Main title<br>Column titles<br>EmptyText-attribute |
| TreeListControl | Main title<br>Column titles<br>EmptyText-attribute |
| TabSetControl | Tabs, Tooltips |
| MenuControl | Main and sub-menu options, Tooltips |

Tabelle 1: Overview of language dependence in case of control elements

| Form elements | Language-dependent components |
|---|---|
| Formular | Main title |
| Formularfelder | Labels, description text, Tooltips |
| Schaltflächen | Possibility for using language-dependent images. |

Tabelle 2: Overview of language-dependence in case of form elements

For internationalizing an application, the Common-Controls Frameworks provides **several options** that can be used either alternately or in combination:

- Configuration of the multi-linguality at application level.
- Configuration of the multi-linguality at session level
- Configuration of the multi-linguality within a JSP page.
- Individual configuration of the multi-linguality for individual control elements and forms

Once the multi-linguatlity has been defined once at the application level, this need not be done again at the level below it, such as a JSP page or within an individual control or form element. If a language treatment is nonetheless defined at the lower level, the settings of the higher-level step are automatically overwritten. This can be meaningful if certain pages always have to be output in a pre-defined language, or if an application is to be converted to multi-linguality in steps.

**The following hierarchy holds good for the language setting:**

- Application Scope
- Session Scope
- Request Scope
- Page Scope
- Control or form element level

# 2 Configuration Options

## 2.1 Configuration of the multi-linguality at application level

The multi-linguality of an application can be configured at application level by storing, in the Servletcontext, an attribute with the key **Globals.LOCALENAME_KEY**. The attribute can be initialized with the following levels:

- **"true"**
  The localization is activated. The Framework uses the local object generated by Struts for language conversions (→ the local object that is stored in the User Session under the key **org.apache.struts.Globals.LOCALE_KEY**).
- **"false"**
  The localization is deactivated explicitly. The setting of a higher level can thus be superseded
- **explicit specification** of a Local Id (Example.: "de" or "en')
  All language conversions are done in the specified language

```
public class FrontController extends ActionServlet {

     /**
      * Constructor for FrontController.
      */
     public FrontController() {
         super();
     }

     /**
      * @see org.apache.struts.action.ActionServlet#init()
      */
     public void init() throws ServletException {

         super.init();

         // Enable resource key translation for all elements
         getServletContext().setAttribute(
             com.cc.framework.Globals.LOCALENAME_KEY, "true");

         // Register all Painter Factories with the favoured GUI-Layout
         // In this case we only use the Default-Layout.
         PainterFactory.registerApplicationPainter(
             getServletContext(), DefPainterFactory.instance());

         PainterFactory.registerApplicationPainter(
             getServletContext(), HtmlPainterFactory.instance());
     }

}
```

CodeSnippet 1: Activation of the multi-linguality at the application level

If a local object is to be explicitly assigned to a user, a corresponding local object can be stored in the session of the user for this purpose. In this manner, individual language support can be implemented. If a local object is to be explicitly assigned to the user, there is a fallback option to a user profile.

The registration of the Local Object is carried out as follows:

```
request.getSession().setAttribute(
    org.apache.struts.Globals.LOCALE_KEY,
    java.util.Locale.ENGLISH);
```

## 2.2 Configuration within a JSP Page.

If the internationalization is to be done only for an individual JSP page, the key **Globals.LOCALENAME_KEY** is saved with the value `"true"` directly in the PageContext of the page. A particular language support can also then be selected.

```
<% pageContext.setAttribute(com.cc.framework.Globals.LOCALENAME_KEY, "true"); %>
```

or

```
<% pageContext.setAttribute(com.cc.framework.Globals.LOCALENAME_KEY, "en"); %>
```

When using templates, it must be remembered that every template gets its own PageContext and this is not passed on in case of Includes. In such cases, the setting must be done separately on all Template pages **or** the setting takes place within the main template, which includes the templates, in which case the key should then be saved in the Request object. The setting then becomes effective to the same extent for all the included pages.

## 2.3 Configuration in case of Control and Form Elements

The language setting can also be made individually for every control element and form element. For activation, the **local**-attribute must be set. The local attribute can, like the key Globals.LOCALENAME, take on the following values:

| Value | Meaning |
|---|---|
| `true` | By specifying `locale="true"` the default Local Object of the user is used in the session (→ stored under **org.apache.struts.Globals.LOCALE_KEY**). |
| `locale` | Allows the direct specification of a Locale Id, like:<br>• locale=”de”<br>• locale=“en“ |

Tabelle 3: Permitted values of the Local-Attribute

# 3 Multi-linguality in Control and Form Elements

## 3.1 Control Elements

In control elements, main titles and column titles as well as the titles of TabPages within TabSets are defined by means of the **title**-attribute. The content that is given there is output in literal form by default (see Code Snippet 1).

```
<ctrl:list
     id="userlist1"
     action="sample101/userBrowse"
     name="users"
     title="User List"
     width="500"
     rows="10"
     refreshButton="true"
     createButton="true">

     <ctrl:columndrilldown   title="Id"         property="userId"       width="65"/>
     <ctrl:columntext        title="Name"       property="name"         width="350"/>
     <ctrl:columntext        title="Role"       property="role.value"   width="150"/>
     <ctrl:columnedit        title="Edit"/>
     <ctrl:columndelete      title="Delete"/>
</ctrl:list>
```

Code Snippet 1: Control element without multilinguality

In the case of an internationalized application, the **title**-attribute is not output as a literal but as a resource key. The key is translated into a concrete character string literal with the help of the country-specific Application Resource properties file. The localization mechanism of Struts is used for the purpose.

```
<ctrl:list
     id="ulist1"
     action="sample101/userBrowse"
     name="users"
     title="userlist1.title"
     width="500"
     rows="10"
     refreshButton="true"
     createButton="true"
     locale="true">

     <ctrl:columndrilldown
             title="userlist1.id"
             property="userId"
             width="65"/>
     <ctrl:columntext
             title="userlist1.name"
             property="name"
             width="350"/>
     <ctrl:columntext
             title="userlist1.role"
             property="role.value"
             width="150"/>
     <ctrl:columnedit
             title="userlist1.edit"/>
     <ctrl:columndelete
             title="userlist1.delete"/>
</ctrl:list>
```

Code Snippet 2: Control element with multilinguality

Then, for supporting the different target languages, only the relevant key/value pairs have to be entered in the resource properties files.

If, for example, an application uses English and German as target languages, the following entries would result for the Code Snippet shown above:

**ApplicationResources_en.properties**

```
userlist1.title=User List
userlist1.id=Id
userlist1.name=Name
userlist1.role=Role
userlist1.edit=Edit
userlist1.delete=Delete
```

**ApplicationResources_de.properties**

```
userlist1.title=Benutzerliste
userlist1.id=Kennung
userlist1.name=Name
userlist1.role=Rolle
userlist1.edit=Bearbeiten
userlist1.delete=Löschen
```

In this example, the ApplicationResources.properties file was configured as a resource file in the struts-config.xml.

```
<struts-config>

      ...

      <!-- Message Resources (Pakage) -->
      <message-resources parameter="ApplicationResources"/>

</struts-config>
```

## 3.2 Multilinguality in Forms

In the case of a localized application, the language-dependent attributes like the **caption**-, **label-** or **title** -attribute of a form are not specified in literal form but also as resource keys. Here too, the conversion into plain text takes place in a manner similar to the control elements.

```
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld"  prefix="html" %>
<%@ taglib uri="/WEB-INF/tlds/cc-base.tld"     prefix="base" %>
<%@ taglib uri="/WEB-INF/tlds/cc-forms.tld"    prefix="forms" %>

<br>

<html:form action="/sample101/userEdit">

    <forms:form
        type="edit"
        caption="frmUserEdit.caption"
        formid="frmEdit"
        locale="true">

        <forms:plaintext
            label="frmUserEdit.id"
            property="userId"/>
        <forms:text
            label="frmUserEdit.lastname"
            property="lastName"
            size="45"
            required="true"/>
        <forms:text
            label="frmUserEdit.firstname"
            property="firstName"
            size="45"
            required="true"/>

        <forms:select
            label="frmUserEdit.role"
            property="rolekey">
            <base:options property="roleOptions"/>
        </forms:select>

        <forms:text
            label="frmUserEdit.email"
            property="email"
            size="45"
            maxlength="256"/>
        <forms:text
            label="frmUserEdit.phone"
            property="phone"
            size="25" />

        <%-- ****************** --%>
        <%-- ** Address      ** --%>
        <%-- ****************** --%>
        <forms:section
            title="frmUserEdit.address">
            <forms:text
                label="frmUserEdit.street"
                property="street"
                size="45" maxlength="80"/>

            <forms:text
                label="frmUserEdit.number"
                property="streetnumber"
                size="5"/>

            <forms:text
                label="frmUserEdit.zipcode"
```

```
                property="zipcode"
                size="5"/>
            <forms:text
                label="frmUserEdit.city"
                property="city"
                size="25"/>

            <forms:select
                label="frmUserEdit.country"
                property="countrycode">
                <base:options property="countryOptions" labelProperty="country"/>
            </forms:select>
        </forms:section>

        <%-- ****************** --%>
        <%-- ** Form Buttons ** --%>
        <%-- ****************** --%>
        <forms:buttonsection default="btnSave">
            <forms:button
                name="btnBack"
                base="images.buttons"
                src="btnBack1.gif"
                title="frmUserEdit.button.cancel"/>
            <forms:button
                name="btnSave"
                base="images.buttons"
                src="btnSave1.gif"
                title="frmUserEdit.button.save"/>
        </forms:buttonsection>
    </forms:form>
</html:form>
```

Then, for the target languages supported, the key/value pairs must be entered in the resource files of the application.

If for example, an application supports English and German as target languages, the following entries would result for the Code Snippet shown above:

**ApplicationResources_en.properties**

```
Images.buttons=app/images/buttons/en

frmUserEdit.caption=User-Edit
frmUserEdit.id=Id
frmUserEdit.name=Name
frmUserEdit.lastname=Last name
frmUserEdit.firstname=First name
frmUserEdit.role=Role
frmUserEdit.email=EMail
frmUserEdit.phone=Phone
frmUserEdit.address=Address
frmUserEdit.street=Street
frmUserEdit.number=Number
frmUserEdit.zipcode=Zipcode
frmUserEdit.city=City
frmUserEdit.country=Country
frmUserEdit.button.cancel=Back
frmUserEdit.button.save=Save
```

**ApplicationResources_de.properties**

```
Images.buttons=app/images/buttons/de

frmUserEdit.caption=Anwender - Editieren
frmUserEdit.id=Id
frmUserEdit.name=Name
```

```
frmUserEdit.lastname=Nachname
frmUserEdit.firstname=Vorname
frmUserEdit.role=Rolle
frmUserEdit.email=EMail
frmUserEdit.phone=Telefon
frmUserEdit.address=Adresse
frmUserEdit.street=Strasse
frmUserEdit.number=Nummer
frmUserEdit.zipcode=PLZ
frmUserEdit.city=Stadt
frmUserEdit.country=Land
frmUserEdit.button.cancel=Abbrechen
frmUserEdit.button.save=Speichern
```

I In this example, the ApplicationResources.properties file was configured in the struts-config.xml as a Resource file.

```
<struts-config>

      ...

      <!-- Message Resources (Pakage) -->
      <message-resources parameter="ApplicationResources"/>

</struts-config>
```

# 4 Multi-Linguality of Buttons

To handle buttons in a language-dependent manner, the **base**-attribute is specified in addition to the **src**-attribute. With the base-attribute, the base directory for the graphic is specified; here too, the localization mechanism can be used – thus, the base directory can be specified as a literal or as a resource key (depending on the current localization setting).

**Example 1 (without Base attribute):**

```
<forms:button
    name="btnSave"
    src="app/images/buttons/btnSave1.gif"
    title="frmUserEdit.button.save"/>
```

The following image is used: app/images/buttons/btnSave1.gif

**Example 2 (Literal base attribute):**

```
<forms:button
    name="btnSave"
    base="app/images/buttons"
    src="btnSave1.gif"
    title="frmUserEdit.button.save"/>
```

The following image is used: app/images/buttons/btnSave1.gif

**Example 3 (Lokalization with a resource key):**

```
<forms:button
    name="btnSave"
    base="images.buttons"
    src="btnSave1.gif"
    title="frmUserEdit.button.save"/>
```

For the following resource settings...

```
ApplicationResources_en.properties

Images.buttons=app/images/buttons/en
```

```
ApplicationResources_de.properties

Images.buttons=app/images/buttons/de
```

… the following images are used

locale="en" : app/images/buttons/en/btnSave1.gif
locale="de" : app/images/buttons/de/btnSave1.gif

# 5 Usage of multiple Resoure Bundels

There are two ways to tell Struts the location of your resource bundle: either by specifying it in your `web.xml` or in the `struts-config.xml` file. You can list multiple `message-resources` tags to load messages from multiple files. If you do this, use the `key` attribute to give a unique name to each bundle. e.g.:

```
<struts-config>

    <!-- Message Resources -->
    <message-resources parameter="ApplicationResources"/>
    <message-resources parameter="MoreApplicationResources"   key="moreResources"/>

</struts-config>
```

You would then have to give the key name when using the `bean:message` tag:

```
<bean:message key="some.message.key" bundle="moreResources"/>
<bean:message key="some.message.key" bundle="moreResources" arg0="1" arg1="2"/>
```

Within the common controls framework you use the following syntax:

**key@resourcebundel#param1#param2**

If only the key is specified, the resource is loaded from the “default” resorce bundel which is stored in the servlet context under the key `org.apache.struts.Globals.MESSAGES_KEY`.

In the following example the tooltip for the back button is loaded form the `MoreApplication Resources.properties` file. The tooltip for the save button comes from the ApplicationResources.

```
<forms:buttonsection>
    <forms:button
        base="images.buttons"
        styleId="btnBack"
        name="btnBack"
        src="btnBack1.gif"
        title="button.title.back@moreResources"/>

<forms:buttonsection>
    <forms:button
        base="images.buttons"
        styleId="btnSave"
        name="btnSave"
        src="btnSave1.gif"
        title="button.title.save"/>
```

# 6 Framework Resource Keys

The framework internal uses some keys to display localized messages.

| Ressource Schlüssel | Message Key | Default |
|---|---|---|
| | | |
| **ListControl** | | |
| FW_ITEMS_NOENTRIES | fw.items.noentries | no entries |
| FW_ITEMS_1TE | fw.items.1te | 1 item |
| FW_ITEMS_ITEMS | fw.items | {0} items |
| FW_ITEMS_RANGE | fw.items.range | {0} to {1} of {2} |
| FW_ITEMS_INFINITE | fw.items.infinite | {0} to {1} of many |
| FW_EMPTY_TEXT | fw.empty.text | No items in list! |
| | | |
| **TreeList** | | |
| FW_PAGE_NOENTRIES | fw.page.noentries | no entries |
| FW_PAGE_1TE | fw.page.1te | page 1 |
| FW_PAGE_RANGE | fw.page.range | page {0} of {1} |
| | | |
| **Tooltips** | | |
| FW_TOOLTIP_CHECKALL | fw.tooltip.checkall | check all items |
| FW_TOOLTIP_CREATE_ITEM | fw.tooltip.create.item | create new item |
| FW_TOOLTIP_FIRSTPAGE | fw.tooltip.firstpage | goto first page |
| FW_TOOLTIP_LASTPAGE | fw.tooltip.lastpage | goto last page |
| FW_TOOLTIP_NEXTPAGE | fw.tooltip.nextpage | goto next page |
| FW_TOOLTIP_PAGE | fw.tooltip.page | goto page {0} |
| FW_TOOLTIP_PREVPAGE | fw.tooltip.prevpage | goto previous page |
| FW_TOOLTIP_REFRESH_LIST | fw.tooltip.refresh.list | refresh list |
| FW_TOOLTIP_UNCHECKALL | fw.tooltip.uncheckall | uncheck all items |
| | | |
| **Gauge** | | |
| FW_EMPTY_GAUGE | fw.empty.gauge | no element |
| | | |
| **Tabbset** | | |
| FW_TABSET_RANGE | fw.tabset.range | {0} ... {1} from {2} |
| | | |
| **Tabbar** | | |
| FW_TABBAR_RANGE | fw.tabbar.range | {0} ... {1} from {2} |

| | | |
|---|---|---|
| **CalendarControl** | | |
| FW_CALENDAR_BUTTON_OK_LABEL | fw.calendar.button.ok.label | Ok |
| FW_CALENDAR_BUTTON_OK_WIDTH | fw.calendar.button.ok.width | 80 |
| FW_CALENDAR_BUTTON_OK_TOOLTIP | fw.calendar.button.ok.tooltip | ok |
| FW_CALENDAR_BUTTON_CANCEL_LABEL | fw.calendar.button.cancel.label | Cancel |
| FW_CALENDAR_BUTTON_CANCEL_WIDTH | fw.calendar.button.cancel.width | 80 |
| FW_CALENDAR_BUTTON_CANCEL_TOOLTIP | fw.calendar.button.cancel.tooltip | cancel |
| FW_CALENDAR_BUTTON_TODAY_LABEL | fw.calendar.button.today.label | Today |
| FW_CALENDAR_WINDOW_TITLE | fw.calendar.window.title | DateTimePicker |
| FW_CALENDAR_WINDOW_WIDTH | fw.calendar.window.width | 350 |
| FW_CALENDAR_WINDOW_HEIGHT | fw.calendar.window.height | 250 |
| FW_CALENDAR_IMAGE_NEXTMONTH_ALT | fw.calendar.image.nextmonth.alt | |
| FW_CALENDAR_IMAGE_NEXTMONTH_TOOLTIP | fw.calendar.image.nextmonth.tooltip | next month |
| FW_CALENDAR_IMAGE_NEXTYEAR_ALT | fw.calendar.image.nextyear.alt | |
| FW_CALENDAR_IMAGE_NEXTYEAR_TOOLTIP | fw.calendar.image.nextyear.tooltip | next year |
| FW_CALENDAR_IMAGE_PREVMONTH_ALT | fw.calendar.image.prevmonth.alt | |

| | | |
|---|---|---|
| FW_CALENDAR_IMAGE_PREVMONTH_TOOLTIP | fw.calendar.image.prevmonth.tooltip | prev. month |
| FW_CALENDAR_IMAGE_PREVYEAR_ALT | fw.calendar.image.prevyear.alt | |
| FW_CALENDAR_IMAGE_PREVYEAR_TOOLTIP | fw.calendar.image.prevyear.tooltip | prev. year |
| FW_CALENDAR_MONTHS | fw.calendar.months | |
| FW_CALENDAR_WEEKDAYS | fw.calendar.weekdays | |
| | | |
| **ColorPicker Control** | | |
| FW_COLORPICKER_WINDOW_TITLE | fw.colorpicker.window.title | ColorPicker |
| | | |
| **Textarea** | | |
| FW_TEXTAREA_MAXLENGTH_MESSAGE | fw.textarea.maxlength.message | Characters remaining: {0}/{1} |

The default messages can be customized. Therefore the localized messages must be registered in the struts.config using the parameter „FrameworkResources“ and the key com.cc.framework.message.

The next example shows, how to configure localized messages in the struts.config

Properties files:

FrameworkResources_de.properties
FrameworkResources_en.properties
FrameworkResources_it.properties

Struts.config (excerpt):

```
<!-- Message Resources -->
<message-resources parameter="ApplicationResources"/>
<message-resources parameter="FrameworkResources"      key="com.cc.framework.message"/>
```