Spring Rich Client

How to deal with the Master/Detail component In Spring Rich Client

Author: Markus Demolsky

1 Overview

The Master/Detail component is still in the sandbox of the Spring Rich Client Project. All components are needed are located in the package: `org.springframework.richclient.form:`

- **`AbstractMasterForm`**
  Abstract base for the Master form of a Master/Detail pair
- **`AbstractDetailForm`**
  Abstract base implementation of the detail side of a Master/Detail form pair
- **`AbstractTableMasterForm`**
  Abstract implementation of AbstractMaster Form that uses a GlazedTableModel and JTable to represent the master information

2 What we need?

2.1 Your master form

Your master form must extend `AbstractTableMasterForm,` only when the master/detail presentation should be a table. The user must override some methods, described in detail now.

2.1.1 getColumnPropertyNames

Tells the component which columns of the details should be displayed in the table. The column headers are read from the message resources.

```
private String[] columnPropertyNames = new
String[] {"publisher",
          "licenseKey",
          "serialNumber",
           "status" };

protected String[] getColumnPropertyNames() {
        return columnPropertyNames;
    }
```

Here is the corresponding message.properties

```
publisher=Publisher of License
licenseKey=Lizenz-Key
serialNumber=Serial-Nr
statusCodes=State
```

2.1.2 createDetailForm

The master form acts also as the container for the detail form. This method creates the detail form which is an instance of **YourDetailForm**, see 2.2.

2.1.3 getMasterCollectionType

The Contract object contains a java.util.List for all child elements (e.g. Licenses of my Contract). This method can now be used to determine the type of the collection, holding the detail items. Especially when working with Hibernate this method **must be** overridden, like this:

```
protected Class getMasterCollectionType(ValueModel collectionPropertyVM) {
        return List.class;
}
```

2.1.4 Sample Master component

```
public class ContractLicenseMasterForm extends AbstractTableMasterForm {
    public static final String BEAN_ID = "contractMasterForm";
    public static final String PAGE_ID = "detailPage";
    // Deklarationen
        private String[] columnPropertyNames = new String[] { "publisher", "licenseKey",
                "serialNumber", "status" };
    private ValidatingFormModel masterModel;

    public ContractLicenseMasterForm(HierarchicalFormModel parentFormModel, String property,
            String formId, Class detailType) {
        super(parentFormModel, property, formId, detailType);
        setSortProperty("license.publisher"); // Via configuration file
        setConfirmDelete(false);
    }

    public void setColumnPropertyNames(String[] names) {
        columnPropertyNames = names;
    }
    /*
     * @see
org.springframework.richclient.form.AbstractTableMasterForm#getColumnPropertyNames()
     */
    protected String[] getColumnPropertyNames() {
        return columnPropertyNames;
    }
    /*
     * @see org.springframework.richclient.form.AbstractMasterForm#createDetailForm
     */
    protected AbstractDetailForm createDetailForm(HierarchicalFormModel parentFormModel,
            ValueModel valueHolder, ObservableList masterList) {
        return new ContractLicenseDetailForm(parentFormModel, "licenseDetail", valueHolder,
                masterList);
    }
    /*
     * @see org.springframework.richclient.form.AbstractForm#createFormControl()
     */
    protected JComponent createFormControl() {
        return super.createFormControl();
    }

    protected Class getMasterCollectionType(ValueModel collectionPropertyVM) {
        return List.class;
    }
```

2.1.5 Set the height of the master table

To set the height of the master table there must be override the following method:

```
protected Dimension getMasterTablePreferredSize(Dimension currentSize) {
    return new Dimension(100,200);  //Your new height of the table
}
```

2.2 Your detail form

The creation of the detail form is straight forward. This detail form is used by the master form when the method **createDetailForm** is invoked.

2.2.1 createControl()

Here is the detail form of the sample:

```
public class ContractLicenseDetailForm extends AbstractDetailForm {
    private HierarchicalFormModel parentModel;

    public ContractLicenseDetailForm(HierarchicalFormModel parentModel,
                                     String arg1, ValueModel arg2,
            ObservableList arg3) {
        super(parentModel, arg1, arg2, arg3);
        this.parentModel = parentModel;
        licenseDao = (LicenseDao)getApplicationContext().getBean(LicenseDao.BEAN_ID);
    }

    /*
     * @see org.springframework.richclient.form.AbstractForm#createFormControl()
     */
    protected JComponent createFormControl() {
        SwingBindingFactory sbf = (SwingBindingFactory) getBindingFactory();
        TableFormBuilder fb = new TableFormBuilder(sbf);
        fb.addSeparator("Lizenzdaten");
        fb.row();
        fb.row();
        fb.add("license.publisher");
        fb.getLayoutBuilder().cell(getLicenseButton());
        fb.add("license.publisherArtTitle");
        fb.row();
        fb.add("license.publisherArtNr");
        fb.add("price");
        fb.row();
        fb.add("licenseCount");
        fb.add("licenseKey");
        fb.row();
        //Some other fields
        //IMPORTANT → CREATE THE BUTTON BAR PROVIDED BY AbstractDetailForm
        fb.getLayoutBuilder().cell(createButtonBar());
        updateControlsForState();
        return fb.getForm();
    }
}
```

2.3 Display the master/detail form

As mentioned before the master form builds the container for the detail form. Therefore the master form is the key for some other work. For instance when you add a form with your created master/detail representation then you must add your master form component.

```
//Some other forms
CompositeDialogPage page = new TreeCompositeDialogPage(ContractGeneralForm.PAGE_ID);
//contract is my Contract POJO
HierarchicalFormModel contractModel = FormModelHelper.createCompoundFormModel(contract);
final ContractGeneralForm generalForm = new ContractGeneralForm(contractModel,
                                                                "generalPage");
//MASTER DETAIL – Important
Form licenseForm = new ContractLicenseMasterForm(contractModel,
                                                 "licenses",
                                                 "detailPage",
                                                  License.class);

page.addForm(generalForm);
page.addForm(licenseForm);

TitledPageApplicationDialog dialog = new TitledPageApplicationDialog(page, page
                                        .getParentWindowControl())
    {
            protected boolean onFinish() {
                return true;
            }
    };
dialog.showDialog();
```

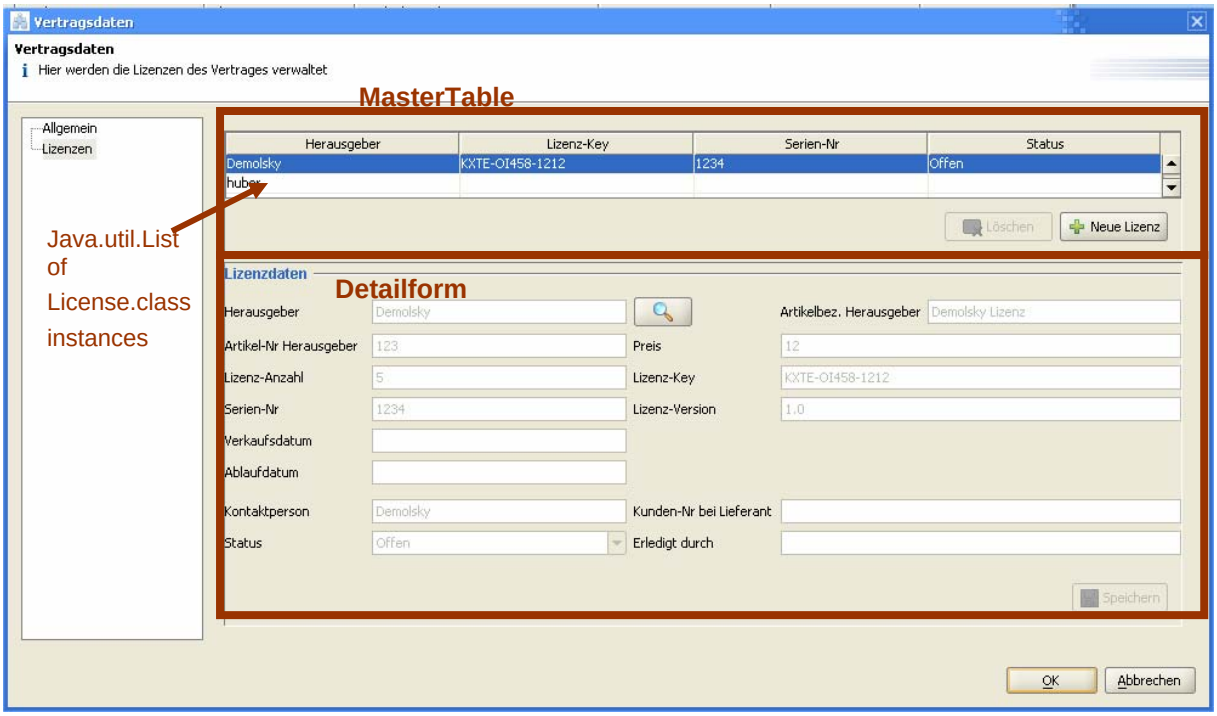
The most important thing is the parameter *"licenses"* that tells the Master/Detail component which property of the Contract POJO contains the details. In our example the Contract saves the details (licenses) in a java.util.List and can be received by *getLicenses()*.

The next important parameter is **License.class**, that tells the Master/Detail component which type of classes are contained in the list.

Here are some further label descriptions (also located in message.properties) that are needed for the button bars:

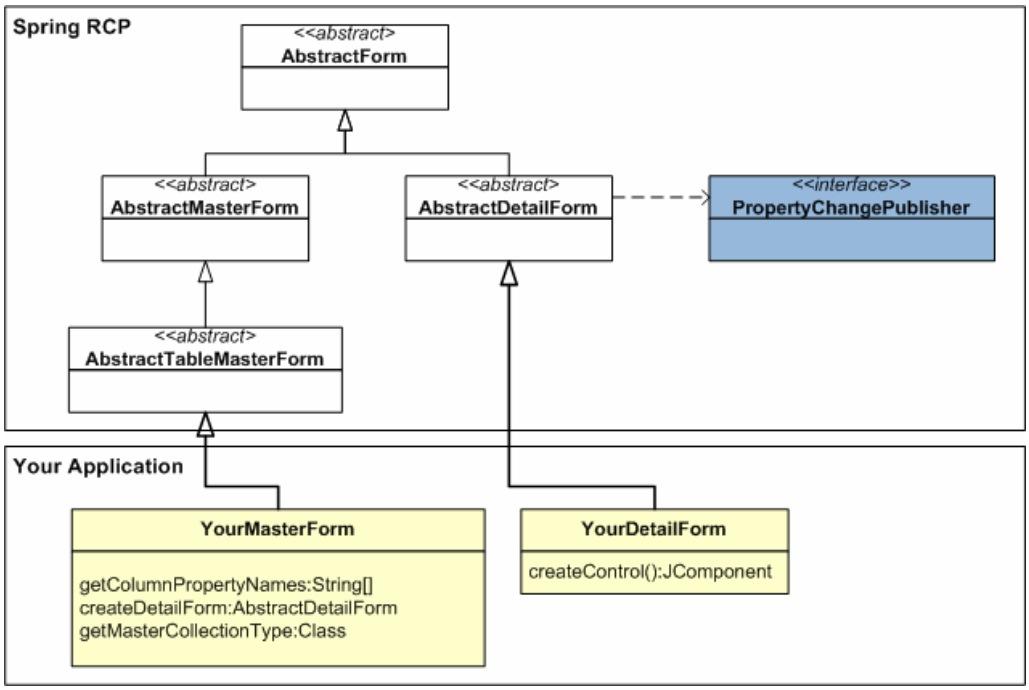
```
save.label=Speichern
revert.label=Revert
cancelNew.label=Cancel
newLicenseCommand.label=Neue Lizenz
deleteDegreeCommand.label=Löschen

//Delete messages
masterForm.dirtyChange.title=Unsaved Changes
masterForm.dirtyChange.message=Selecting a different item will cause you
to lose your unsaved changes.\nAre you sure you want to select a different
item?

masterForm.dirtyNew.title=Unsaved Changes
masterForm.dirtyNew.message=Creating a new item will cause you to lose
your unsaved changes.\nAre you sure you want to do this?

masterForm.confirmDelete.title=Confirm Delete
masterForm.confirmDelete.message=Are you sure you want to delete these
items?
```

Screenshot:


Vertragsdaten
Vertragsdaten
Hier werden die Lizenzen des Vertrages verwaltet
MasterTable
Allgemein
Lizenzen
Herausgeber
Lizenz-Key
Serien-Nr
Status
Demolsky
KXTE-OI458-1212
1234
Offen
huber
Java.util.List of License.class instances
Löschen
Neue Lizenz
Lizenzdaten
Detailform
Herausgeber
Demolsky
Artikelbez. Herausgeber
Demolsky Lizenz
Artikel-Nr Herausgeber
123
Preis
12
Lizenz-Anzahl
5
Lizenz-Key
KXTE-OI458-1212
Serien-Nr
1234
Lizenz-Version
1.0
Verkaufsdatum
Ablaufdatum
Kontaktperson
Demolsky
Kunden-Nr bei Lieferant
Status
Offen
Erledigt durch
Speichern
OK
Abbrechen

3 UML Representation

The master detail component consists of the following parts:



`AbstractMasterForm` is the master form of a master/detail pair. In the above domain model the contract is the master.

`AbstractDetailForm` is the detail form of a master/detail pair. In the above domain model the license is the detail.

Both `AbstractMasterForm` and `AbstractDetailForm` subclasses from `AbstractForm`, ensuring that these forms can be displayed in a container, like a CompositeDialogPage.

In most cases the presentation of a master/detail relationship is represented as a table. Therefore the `AbstractTableMasterForm` exists. This implementation displays the details in a table, including sort, property change listening and other features.

When adapting the Master/Detail component in applications, `AbstractTableMasterForm` and `AbstractDetailForm` are the important extensions.