# Eclipse Scout

## *Release Notes*

Scout Team

Version 8.0

# Table of Contents

# About This Release

Eclipse Scout 8.0 will be part of the Eclipse *Photon* release. It will be released in June 2018 (release schedule). The latest version of this release is: None released yet.

**Attention:** The here described functionality has not yet been released and is part of an upcoming release.

- [PLACEHOLDER]

You can see the detailed change log on GitHub.

# Service Releases

Please note that with the Eclipse Photon, there will no longer be regular *service releases*, but instead a faster release cycle. See the Simultaneous Release Cycle FAQ for details.

# Obtaining the Latest Version

## Runtime (Scout RT)

Scout RT artifacts are distributed via Maven:

- 7.1.0.007_Photon_M6 on *Maven Central*
- 7.1.0.007_Photon_M6 on *mvnrepository.com*

Usage example in the parent POM of your Scout application:

```
<dependency>
    <groupId>org.eclipse.scout.rt</groupId>
    <artifactId>org.eclipse.scout.rt</artifactId>
    <version>7.1.0.007_Photon_M6</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

## Eclipse IDE Tooling (Scout SDK)

You can download the complete Eclipse IDE with Scout SDK included (EPP) here:
Eclipse for Scout Developers

To install the Scout SDK into your existing Eclipse IDE, use this update site:
http://download.eclipse.org/scout/releases/8.0/7.1.0/007_Photon_M6/

## Demo Applications

The demo applications for this version can be found on the features/version/7.1.0.007_Photon_M6 branch of our docs repository on GitHub.

If you just want to play around with them without looking at the source code, you can always use the deployed versions:

- https://scout.bsi-software.com/contacts/
- https://scout.bsi-software.com/widgets/
- https://scout.bsi-software.com/jswidgets/

# Java 8 required

The required Java Runtime Environment (JRE) to run an Eclipse Scout application has changed: Starting with Eclipse Scout 8.0, a **Java 8 runtime is required**.

The Scout 8.0 Runtime does not support Java 9 or Java 10 yet. The support for newer Java versions is planned for a later release.

# New browser version requirements

The supported browser versions have been updated. See https://eclipsescout.github.io/8.0/technical-guide.html#browser-support for the new list of supported browsers.

# New SDK Feature in Eclipse: Search for missing NLS keys

If NLS keys are used in the code that do not exist in a properties file, an ugly placeholder is displayed to the user. To find such missing translations the new Menu `Scout → Search missing text keys⋯` may be handy. The result is listed in the Eclipse `Search` view.

The search also takes the scope of each NLS key into account. So that the key is considered to be available there must be a `TextProviderService` with that key on the classpath of that module.

Reported false positives can be suppressed using the following comment at the end of the corresponding line: `NO-NLS-CHECK`. Matches on that line are then not reported in future searches anymore.

# Config Properties

## Descriptions

Config properties based on `org.eclipse.scout.rt.platform.config.IConfigProperty` include a description text. This description is stored in the new `description()` method.

The class `org.eclipse.scout.rt.platform.config.ConfigDescriptionExporter` can be used to export these descriptions. By default an AsciiDoctor exporter is included.

All Scout properties have been extended with descriptions. The same text is also part of the technical documentation.

## Default Value

Config properties based on `org.eclipse.scout.rt.platform.config.IConfigProperty` include a default value. The default value is stored in the `getDefaultValue()` method.

The method was moved from `org.eclipse.scout.rt.platform.config.AbstractConfigProperty<DATA_TYPE, RAW_TYPE>` to the interface. Therefore the visibility has changed from protected to public.

## Validation

The concrete implementation `org.eclipse.scout.rt.platform.config.ConfigPropertyValidator` which validates the configuration of `config.properties` files will also check if a configured value matches the default value. In case it does a info message (warn in development mode) will be logged but platform will still start. To minimize configuration files such entries should be removed from `config.properties` files.

# GroupBox Enhancements

## Layout Configuration

It is now possible to adjust the parameters of how the group box will be layouted. The following parameters may be set:

**hgap**
: the horizontal gap in pixels to use between two logical grid columns

**vgap**
: the vertical gap in pixels to use between two logical grid rows

**columnWidth**
: the width in pixels to use for a grid column

**rowHeight**
: the height in pixels to use for a grid row

**minWidth**
: the minimum width of the group box. If this width is > 0 a horizontal scrollbar is shown when the group box gets smaller than this value.

These values may be set using `getConfiguredBodyLayoutConfig()`.

## Sublabel

GroupBoxes got a new property called `sublabel`. The sublabel is displayed below the title in a very small font.

## Notification

Add a `INotification` to a group box with the new property called `notification`.
Use `IGroupBox.setNotification(INotification)`, `getNotification()`, `removeNotification()` to control it.
A notification has a `IStatus` which includes a severity and a message.

By default the notification is displayed at the beginning of the group box body.

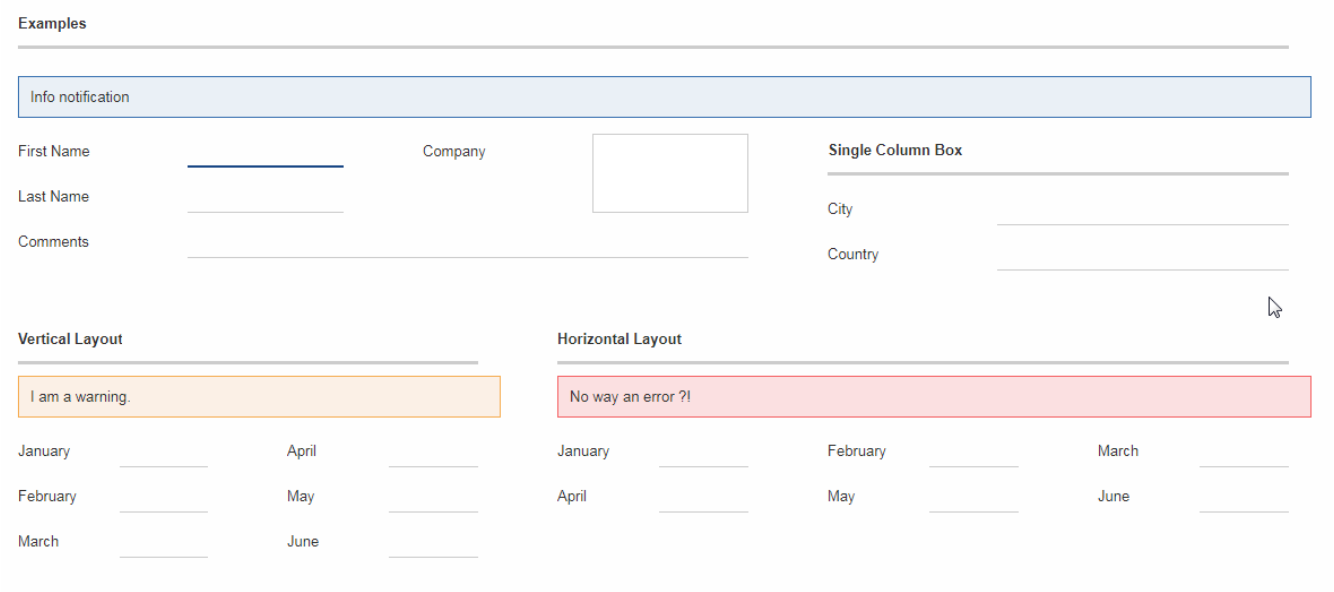


*Figure 1. Group box notification*

A demo may be found here: https://scout.bsi-software.com/widgets/?dl=widget-groupbox.
And here for the JS only version: https://scout.bsi-software.com/jswidgets/#groupbox.

# Tree Traversal Engine Added

A tree (or graph) traversal engine has been added to the Scout Platform. The engine can traverse any tree- or graph-like structures. See the package `org.eclipse.scout.rt.platform.util.visitor` especially the class `org.eclipse.scout.rt.platform.util.visitor.TreeTraversals` as entry point.

```
TreeTraversals.create(visitor, childrenSupplier).traverse(root);
```

The `root` element can be of any type. The `childrenSupplier` is a Function that returns the child elements for a given parent. The visitor can be a `IBreadthFirstTreeVisitor` for Breadth-First visiting strategy or a `IDepthFirstTreeVisitor` for a Depth-First visiting strategy supporting pre-order and post-order traversals.

# Introducing Widget.java

On JavaScript side, there has been a class `Widget.js` for a long time now. With this release the counterpart `Widget.java` has been added. This gives all existing widgets like `FormField`, `Form`, `MessageBox`, `Action` (e.g. `Menu`), `Tree`, `Table`, `Accordion`, `Calendar`, `Desktop`, `Tile`, etc. a new common base class. It also helps creating widgets which aren't necessarily form fields.

The new Widget class handles the widget lifecycle (initConfig, init, disposed) and offers methods to visit Widget hierarchies. See `org.eclipse.scout.rt.client.ui.IWidget` for more details.

# New Widget 'TileGrid'

The new `TileGrid` widget arranges `Tile` s in a grid by using the `LogicalGridLayout`. This is the same layout as used for a `GroupBox`, so the same `GridData` object may be used to configure how the individual tiles should be arranged.

A `Tile` directly extends `Widget` and is not much more than a `<div>` with the CSS class `tile`. In order to customize your tile you have to create a custom widget, which is easier than it sounds. Just create a JS class lets say `CustomTile.js` which extends from `Tile.js`, create a Java class `CustomTile.java` which extends from `AbstractTile.java` and add some glue code to link them together. See the code of the demo widgets on GitHub for details. You could also use existing widgets as tiles. In that case instead of extending `AbstractTile` you would extend `AbstractWidgetTile` or `AbstractFormFieldTile` and set the property `tileWidget` accordingly.

In order to add the `TileGrid` to a form, you can use the class `TileField` which is basically a simple `FormField` wrapping the `TileGrid`. You cannot use the `TileGrid` directly because a `GroupBox` only accepts `FormField` s.

A demo of the widget may be found here: https://scout.bsi-software.com/widgets/?dl=widget-tilefield.

And here for the JS only version: https://scout.bsi-software.com/jswidgets/#tilefield.

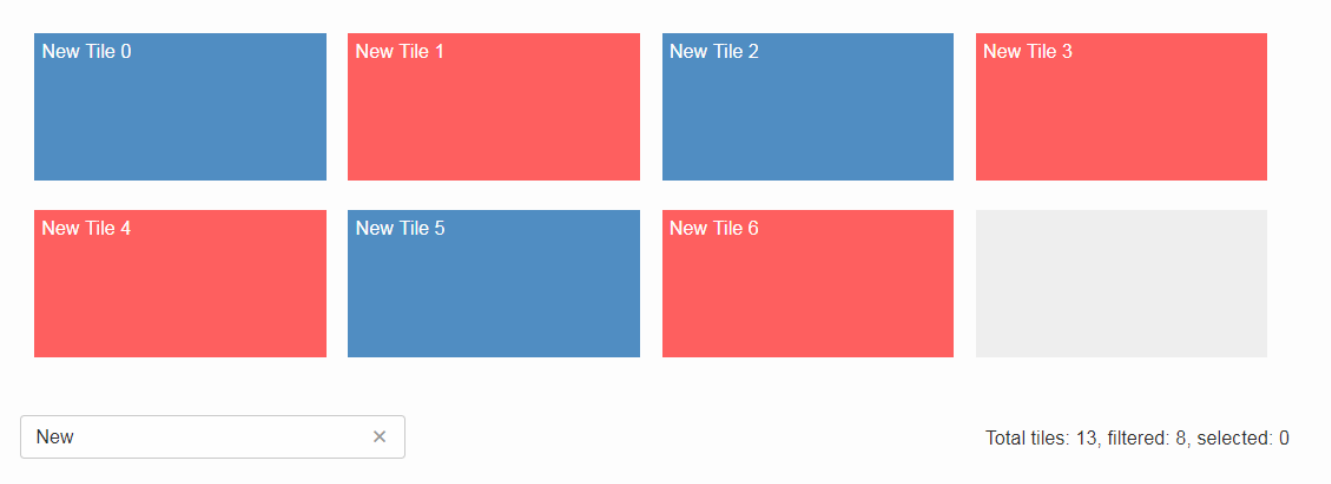


*Figure 2. TileGrid*

# New Widget 'Accordion'

The `Accordion` displays several collapsible `Group` s. The default behavior is to collapse every other group if one group is expanded. Because that is not in any case desired, the behavior may be disabled by setting the property `exclusiveExpand` to false.

The `Group` is a simple widget containing of a header and a body. The body may be any other widget like the new `TileGrid`. Because having tiles in an accordion is a typical use case, there is a widget called `TileAccordion` which helps creating the groups and provides some delegate methods to easily access the tiles of every group. It also takes care that selecting multiple tiles across the individual groups works as there were only one single `TileGrid`.

A demo of the accordion may be found here: https://scout.bsi-software.com/widgets/?dl=widget-accordionfield.
And here for the JS only version: https://scout.bsi-software.com/jswidgets/#accordion.

A demo of the tile accordion may be found here: https://scout.bsi-software.com/widgets/?dl=widget-tileaccordionfield.
And here for the JS only version: https://scout.bsi-software.com/jswidgets/#tileaccordion.

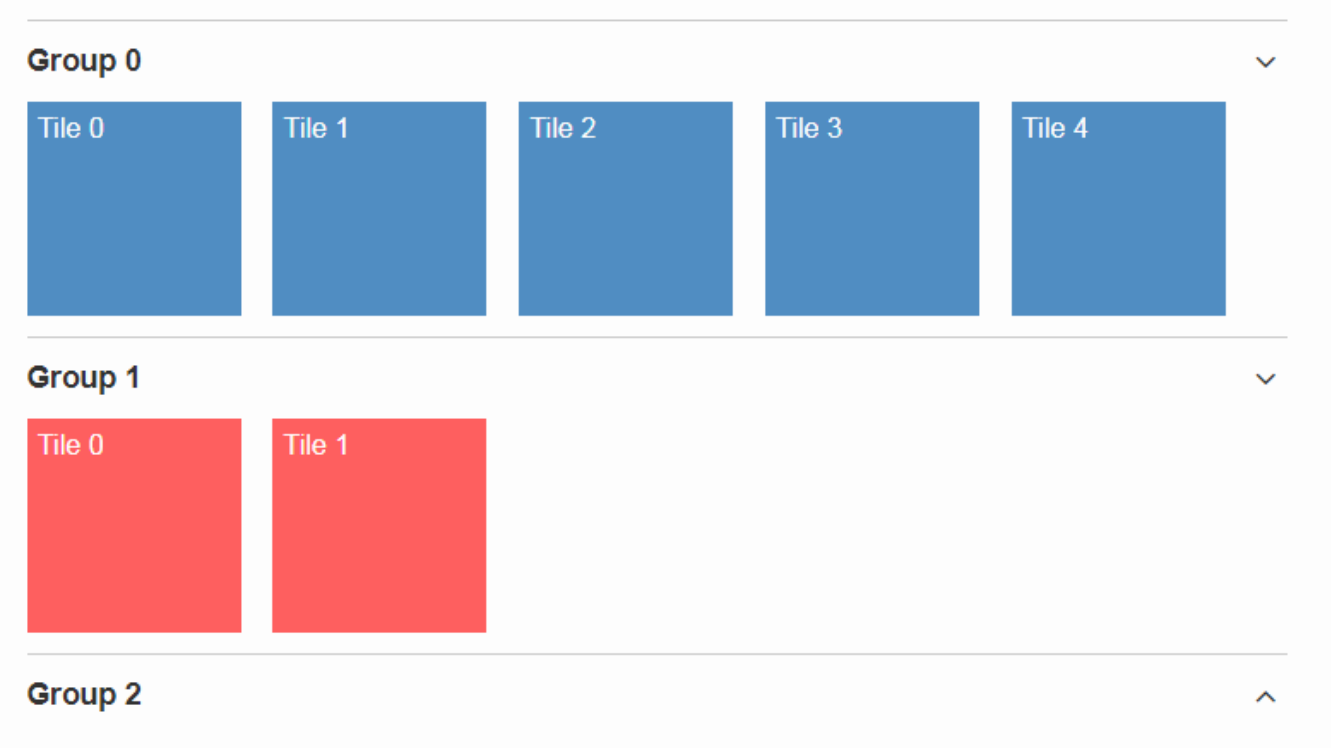


*Figure 3. Accordion*

# New Outline Overview

The `OutlineOverview` typically is the first thing a user sees when an outline based application starts. It is the widget displayed in the desktop bench when no page of the outline is selected. The previous `OutlineOverview` has been very simple, it basically just displayed the title and the icon of the outline. With this release a new widget has been added, it is called `TileOutlineOverview`.

As the name implies it is based on the new `TileGrid` and shows the top level pages of the current outline. The tiles itself are very simple: they show the name of the page and an icon. Because they look a lot more interesting with an icon we encourage you to set a distinct icon for each page. You can do this by using the property `overviewIconId` of a page. Note that this is not the same property as `iconId`, to be able to use different icons for the outline tree itself and the outline overview tiles.

The new outline overview is enabled by default. If you don't like the new style you can either use a `defaultDetailForm` which will be used instead of the `outlineOverview` if it is set. Or you can use the old `OutlineOverview` or even create a custom one. Compared to `defaultDetailForm`, the `outlineOverview` can only be set using JavaScript.

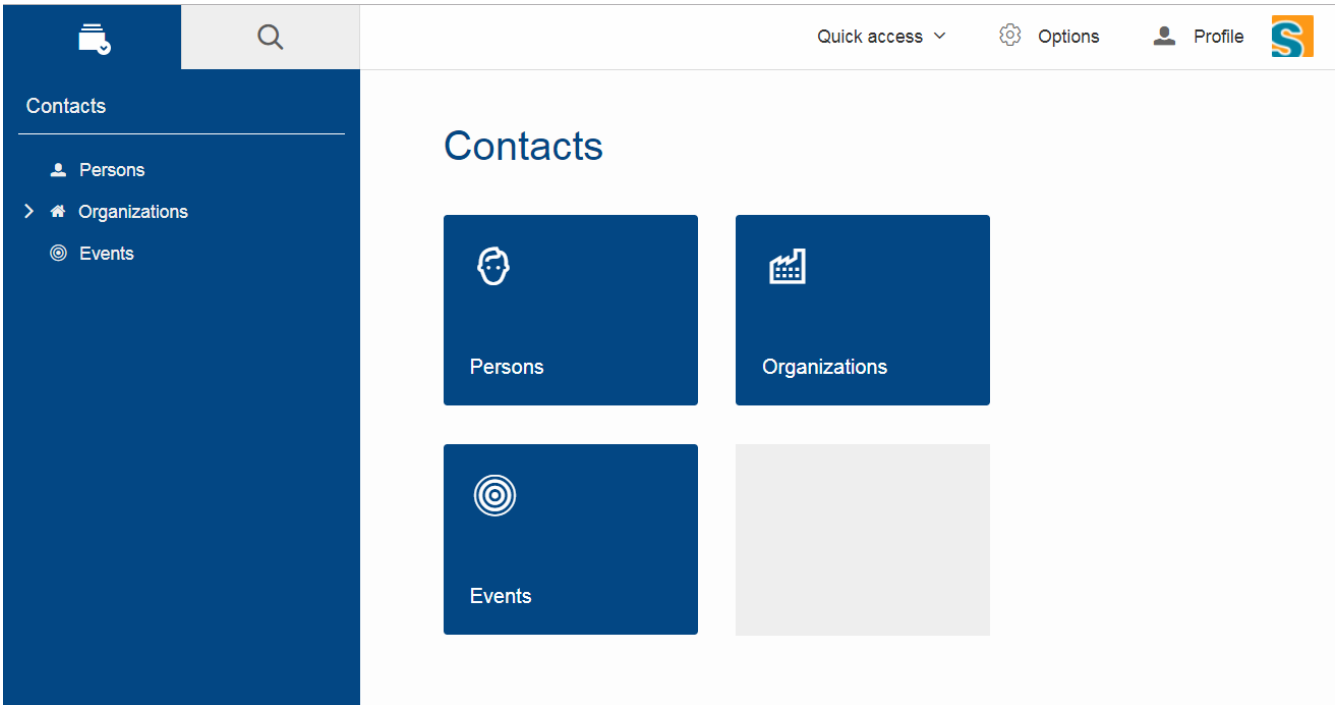


*Figure 4. Tile Outline Overview*

# Menu and MenuBar Enhancements

## Form Field in Menu

The menubar now supports form field menu items (`FormFieldMenu`). On the model side extend `AbstractFormFieldMenu` with a form field as an inner class to use a form field menu in any menu supporting environment.

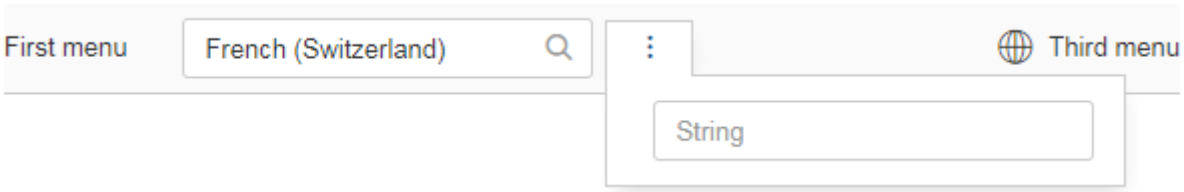


*Figure 5. Menubar with form fields*

## New Property 'stackable'

The menu property `stackable` defines if a menu is stackable or not. A stackable menu will be moved to the ellipsis dropdown menu when there is not enough space in the menubar. The ellipsis menu is placed after the last stackable menu in the menubar. Right and left aligned menus will be moved to a single ellipsis menu per menubar. The horizontal alignment of the ellipsis menu is the same as the last stackable menu in the menubar.

# TabBox Enhancements

## Left Aligned Menu Items

The menubar of a tabbox now considers the menu alignments *left* and *right*. That means you can add menus directly on the right side of the last tab item (left aligned) or at the right side of the tab box header (right aligned).

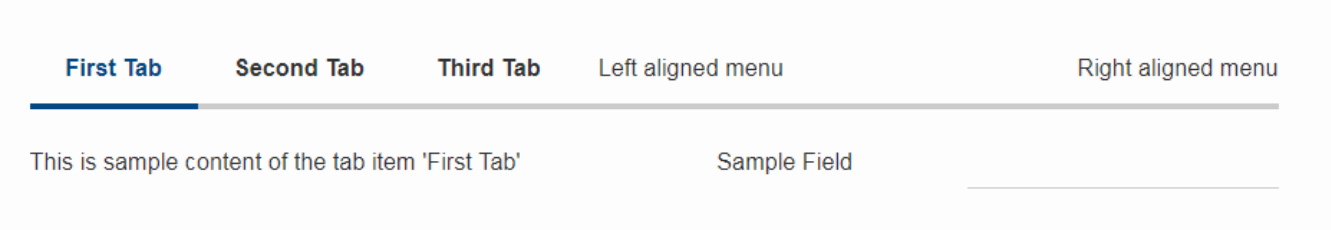


*Figure 6. Menus in a tab box header*

## Collapsible Menu Items

Menus in the menubar will be moved to an ellipsis menu in case there is not enough space in the tabbox header. The tab items are moved to an ellipsis menu when there is not enough space for all tabs. The collapse order is as following: all menus are collapsed first before the tabs will be collapsed from right to left.

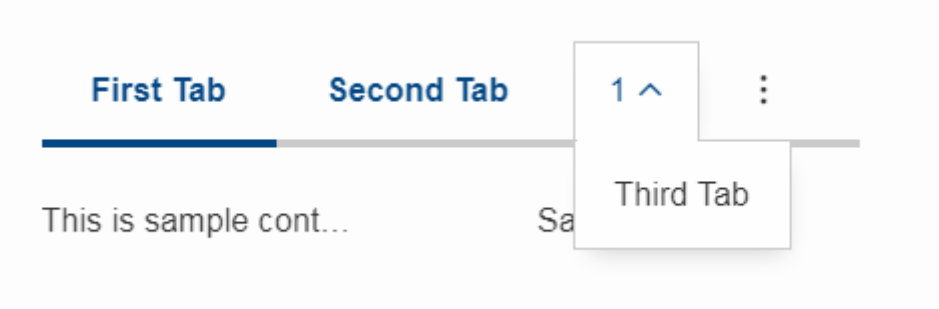


*Figure 7. Ellipsis menu for the tabs of a tab box*

## Sublabel

TabItems got a `sublabel` property which is displayed in a very small font below the title (see also Sublabel).

## Animated Selection Marker

The marker of the selected tab is now animated and follows the user or model selection.

## Optimized Zoom Behavior

Several bugfixes of pixel issues due to zoom levels.

# Desktop Splitter Position Remembered Across Sessions

The position of the desktop splitter position (between the navigation and the bench) is now persisted across sessions, i.e. the previous setting will be restored even after you closed your browser. The position is stored in the HTML 5 *local storage* provided by the local browser. It is therefore a device-specific setting, which is especially useful when accessing the same application through screens with different resolutions.

In case the splitter position should never be remembered, the feature can be disabled globally by setting the property `cacheSplitterPosition` on the desktop to *false*.

# ImageField: Support for SVG Images and Image URLs

It's now possible to use SVG images in the same way as bitmap images. Simply put the .svg file in the `/icons` folder of the client module and reference the SVG image in any widget that supports the `iconId` property. Example:

```
@Override
protected String getConfiguredIconId() {
  return "person.svg";
}
```

Additionally you can now reference an image by URL, for instance an image hosted on an external server. Use the property ´imageUrl` of the AbstractImageField to reference the image. Note: the AbstractImageField defines a priority for which one of the three image properties is used to render the image in the browser:

1. *image* (Binary resource)
2. *imageUrl*
3. *imageId*

# CheckBoxField: Add Support for Key Strokes

The check box field (`IBooleanField`) got a new property called `keyStroke`. The property expects a string defining the key stroke, e.g. `ctrl-b`. When the key stroke is executed the check box value will be toggled. Other widgets like `Button`, `RadioButton` or `Menu` already support that feature in the same way.

# Layout Customizations for Radio Button Group

A new property `gridColumnCount` has been added to the radio button group. It can be used using `setGridColumnCount()`, `getGridColumnCount()` and `getConfiguredGridColumnCount()`. By default the columns are configured to be dependent on the height of the field to create columns as needed to show all radio buttons within the height available (this also corresponds to the existing behavior).

But it also allows to specify an exact number over how many columns radio buttons should be distributed. This is an alternative to layout the buttons using the group height and is especially useful if the number of radio buttons is unknown or dynamic. In that case the columns can be configured to e.g. 3 and the property `useUiHeight` to true allowing the group to vertically grow as needed to show all radio buttons within 3 columns. This property also corresponds to the layout possibilities of the group box.

The same possibilities also exist in the JavaScript only layer of Scout using the method `setGridColumnCount()`.

# Dynamic Fields

It is now possible to add and remove fields dynamically also when a form is already started. This feature is supported for `GroupBoxes` and `TabBoxes`.

The Java API orders the added fields considering the `order` member.

API:

- TabBox.js `insertTabItem`, `deleteTabItem`, `setTabItems`
- GroupBox.js `setFields`, `insertField`, `insertFieldBefore`, `deleteField`
- ICompositeField.java `setFields` and the already existing `addField`, `removeField` methods which don't throw an exception anymore when a form is already initialized.

- The support for adding ProcessButtons dynamically is not implemented so far.
- Adding a field to container (`TabBox`, `GroupBox`) forces the container to be rendered. All fields in this container will be removed and rendered again.

# Form Fields

## New Field Style

This release introduces a new field style called `alternative`. This is the new default style for every form field. The `classic` style is still available because it may be preferable in some circumstances, e.g. when used in a cell editor or on a form with background color like the search form. For these two cases the style is set to `classic` automatically but you can do it for your custom cases as well by setting the new property `FieldStyle`.

If you want to revert your whole application to the classic style you can create an extension to `AbstractFormField` and change the default of the `FieldStyle` property. For Scout JS applications you can set the variable `scout.FormField.DEFAULT_FIELD_STYLE` to `scout.FormField.FieldStyle.CLASSIC`;

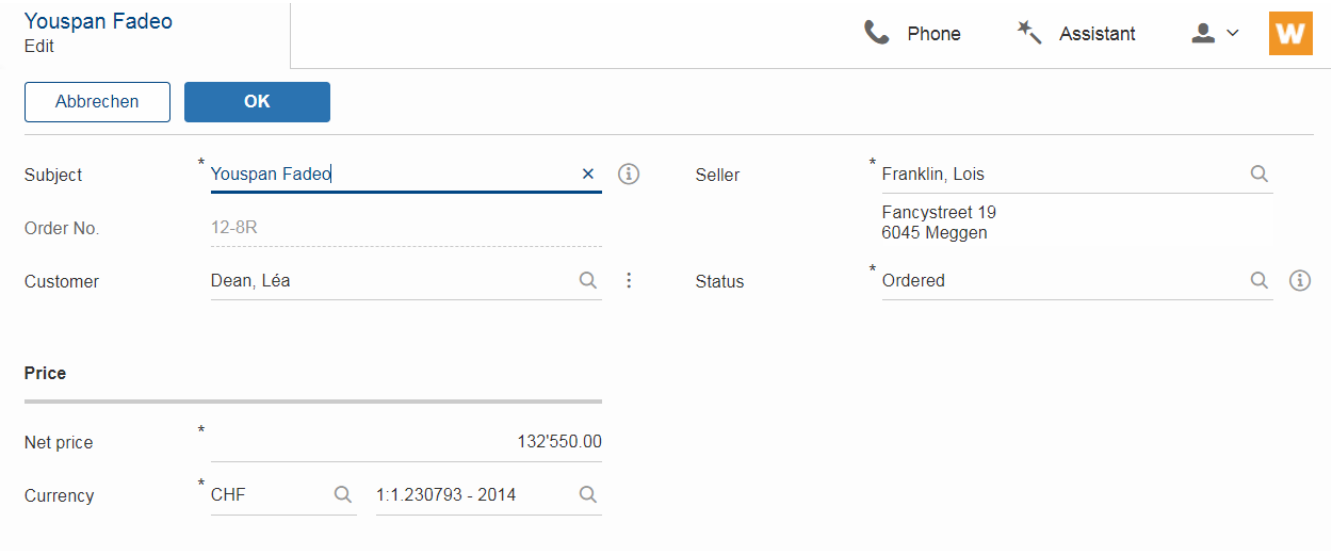


*Figure 8. New alternative field style*

## Improved Accessibility

The label and the input are now linked by using `aria-labelledby`. This allows screen readers to read the label if an input is focused.

Furthermore, clicking the label will now activate the field. This is especially helpful on mobile devices when the new alternative style is active, because the field boundaries are not obvious anymore.

# Enhanced IUiServletRequestHandler

UI Servlet request handler now supports all HTTP methods and not only GET and POST. When using `AbstractUiServletRequestHandler` no migration should be required, see migration guide for further information.

# Automatic Preloading of Web Fonts

To prevent incorrect measurements or the so-called "FOUT effect" (Flash Of Unstyled Text), Scout tries to preload all necessary web font files (*.woff) before rendering the application. To make it easier for projects to add theme-dependent fonts, the font preloader has been improved. The list of fonts to preload is now detected automatically by inspecting the document's style sheet (*@font-face* rules). It's no longer necessary to manually list all fonts in the bootstrap argument of `scout.App` (see migration guide).

# Improved useUiHeight Calculation

If a form field is set to use its ui height, it is supposed to be as big as its content. A typical example is the group box: `useUiHeight` is true by default to make the group box as height as the containing form fields. In that case it works fine because the height does not depend on the width.

There are cases where the height depends on the width, e.g. if a label field is set to wrap its text (property `wrapText` = true). These cases did not work correctly because in order to calculate the preferred height the final width has to be known. This has been fixed so that setting `useUiHeight` to true should now work as expected.

# Refactored EventListenerList

The class *EventListenerList* had poor performance with large numbers of listeners and add / remove operations. There are two new alternatives to this class that are also thread-safe and support higher performance:

- `FastListenerList<LISTENER>` is used to manage a single type of listeners.
- `AbstractGroupedListenerList` is used as base class to handle a single type of listener with multiple type partitions. See the new classes TreeListeners and TableListeners for an example of applicability.

Consider refactoring the use of *EventListenerList* by one of the new alternatives.

# New Property 'searchRequired' on Smartfields

A new property `searchRequired` has been introduced for Smartfields. It is similar to the one already existing in `org.eclipse.scout.rt.client.ui.desktop.outline.pages.AbstractPageWithTable` and controls the Smartfield behavior if the proposal-list is opened without having a search constraint. By default (*searchRequired* = `false`) all existing proposals are shown if no search constraint has been typed. But if the property is set to `true`, the Smartfield only shows proposals if a search constraint is available. This is especially useful if a large data set is expected in a Smartfield lookup which usually makes no sense to present all to the user. In that case a message is shown instead informing that a search constraint is required to load data and to see proposals.

In Java the property can be set using `ISmartField.setSearchRequired()` or `AbstractSmartField.getConfiguredSearchRequired()`. In JavaScript the property can be set using `smartfield.setSearchRequired()`.

# New Event 'prepareLookupCall' on Smartfields and TagFields

In Scout JS a new event `prepareLookupCall` has been added to the SmartField and TagField. It allows to be notified when the field is about to execute a LookupCall. Because for each call a fresh LookupCall clone is executed this event allows to propagate properties to the executing LookupCall clone. These properties may then be used when the call is executed (e.g. sent to the backend).

# Minimum and maximum values can be set for NumberFields in Scout JS

The Scout JS NumberField now also supports min- and max-values as it was already present in Scout Classic.

# Hierarchical Table Support

Scout now supports hierarchical tables. The property `parentKey` on `IColumn` is responsible for the linking between parent and child rows. The property `hierarchicalStyle` on the table is used to switch between the default or structured style.

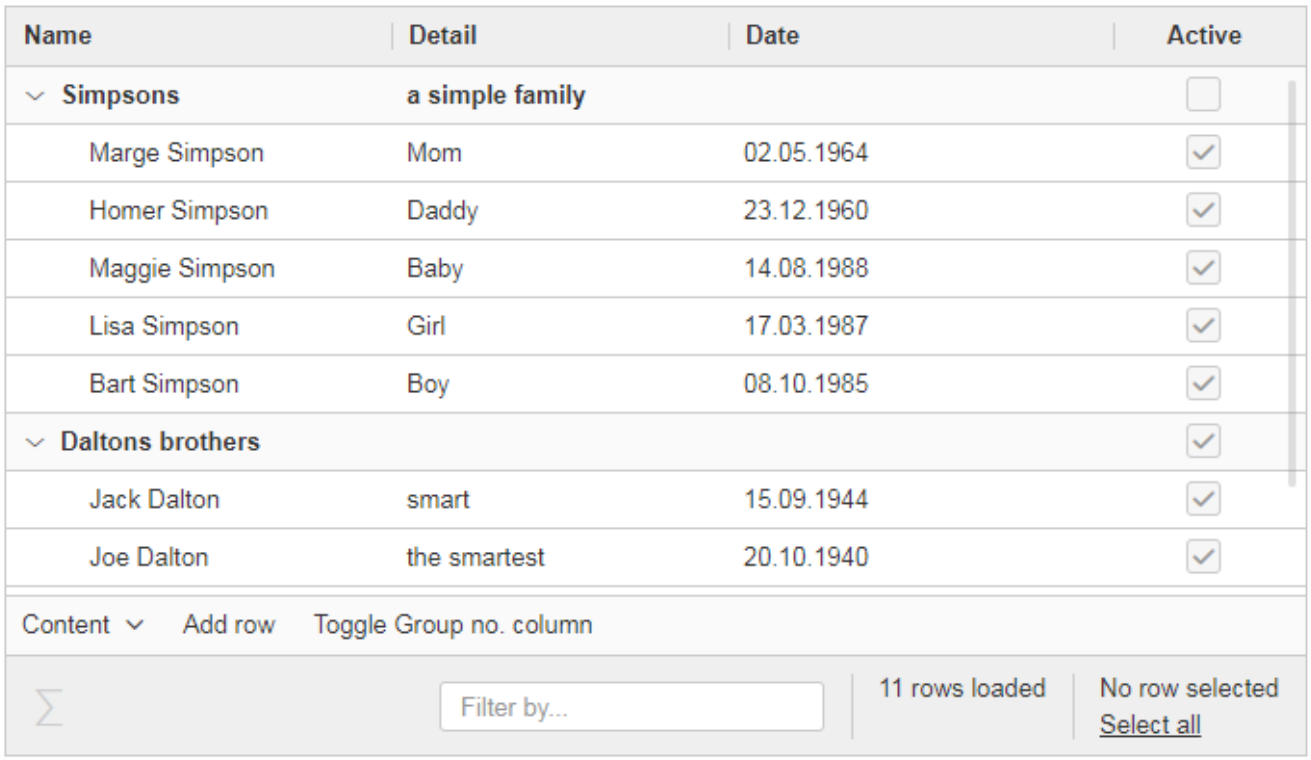


*Figure 9. Hierarchical Table Example*

# Text and Multi-Language Support Moved to 'platform'

Scout's multi-language text support mainly consists of `ITextProviderService` (with the default abstract implementation `AbstractDynamicNlsTextProviderService`) and the convenience accessor `TEXTS`.

This facility was moved from the module `org.eclipse.scout.rt.shared` to `org.eclipse.scout.rt.platform`. This allows the use of multi-language text support without the need to include **.shared* dependencies in your project. This is especially useful for server-only applications (e.g. micro services).

*Migration:* Change the `import`s in your **.java* files accordingly.

Do you want to improve this document? Have a look at the sources on GitHub.