

Eclipse Scout

Release Notes

Scout Team

Version 9.0

Table of Contents

About This Release	1
Service Releases	1
Obtaining the Latest Version	1
Dark Theme	3
New Servlet Filters to Create a Scout RunContext	4
New Widgets	5
Mode Selector	5
Popup	5
Label	6
Disabling Close- & Cancel-Buttons	7
Improved Scrollbar Usability.....	8
Design Change for WizardProgressField	9
Improvements for Pages in Scout JS Applications	10
New Event "lookupCallDone"	11
Property Lookup Order Changed	12
New CheckableStyle for Table and Tree	13
Strings Sorted with "Natural" Collator by Default	14
New Properties for MenuBar Design	15
New GroupBox Property 'menuBarPosition'	15
New GroupBox Property 'menuBarEllipsisPosition'	15
New Menu/Button Property 'shrinkable'	15
New Button Property 'stackable'	15
New OpenUriAction	17
New Column Property 'nodeColumnCandidate'	18

About This Release

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

The Eclipse Scout 9.0 version is planned to be part of the Eclipse 2019-03 Simultaneous Release ([release schedule](#)). The release is scheduled for March 2019.

The latest version of this release is: (not yet released)

You can see the [detailed change log](#) on GitHub.

Service Releases

After Eclipse Photon, there no longer are Eclipse *service releases* (see [the Simultaneous Release Cycle FAQ](#) for details).

There may be additional maintenance builds of Scout 9.0, but no plans have been finalized yet. Beside bugfixes, these maintenance builds may even contain some minor features. See the following notes for details.

Simrel 2019-06 (9.0.1) Release Expected in June 2019

(Section intentionally left blank for possible future release)

Obtaining the Latest Version

Runtime (Scout RT)

Scout RT artifacts are distributed via Maven:

- [8.0.0.021_Simrel_2018_09](#) on *Maven Central*
- [8.0.0.021_Simrel_2018_09](#) 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>8.0.0.021_Simrel_2018_09</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:

To install the Scout SDK into your existing Eclipse IDE, use this update site:

http://download.eclipse.org/scout/releases/8.0/8.0.0/018_Simrel_2018_09_M3/

Demo Applications

The demo applications for this version can be found on the [features/version/8.0.0.021_Simrel_2018_09](#) 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/>

Dark Theme

Enter the dark side... and use the new dark theme of Scout!

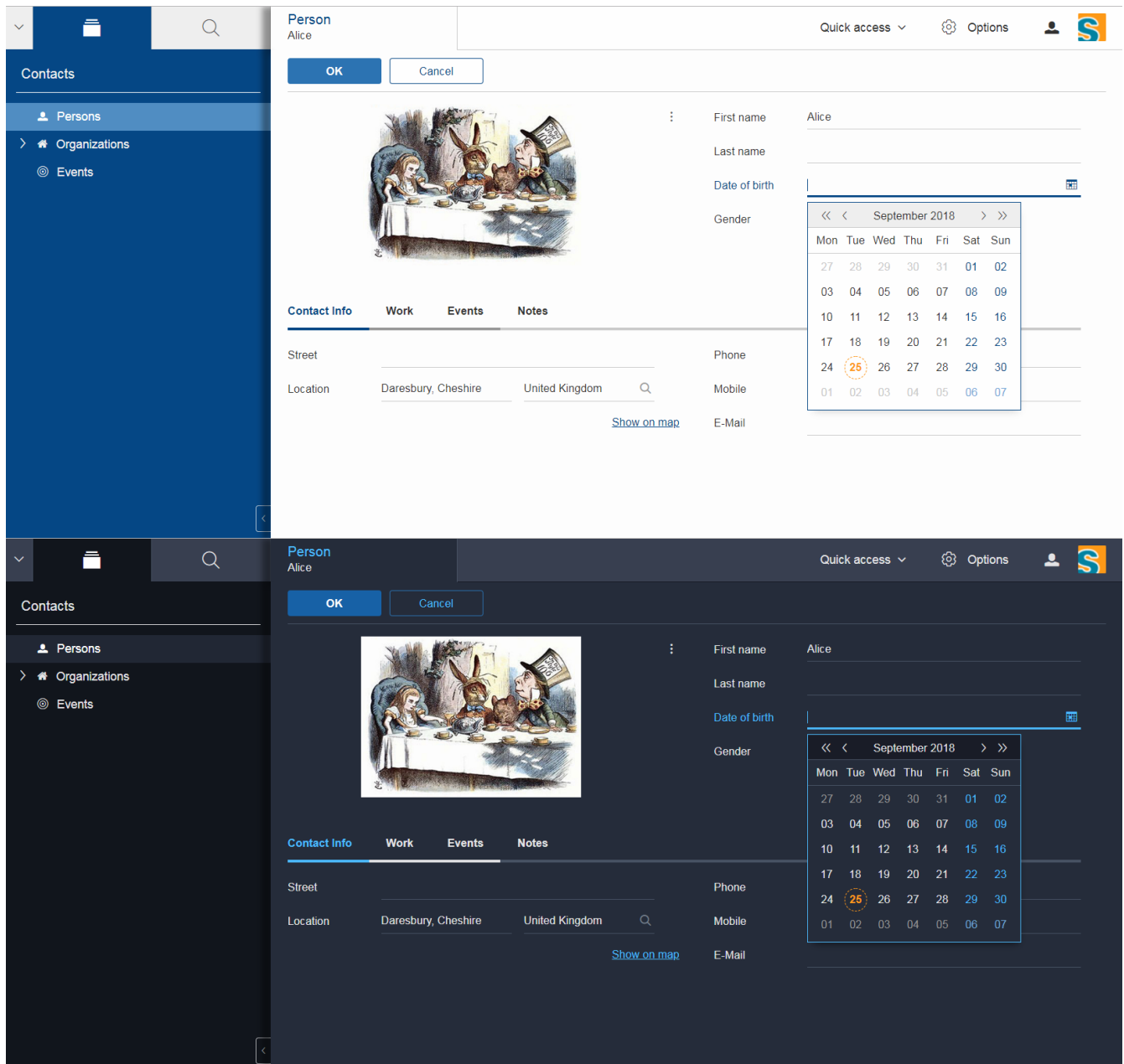


Figure 1. Dark Theme

Scout now provides a dark theme in addition to the default theme. You can either activate it by default by setting the property `scout.ui.theme` to `dark` in the `config.properties`, or let the user choose what he likes more.

In order to change the theme during runtime you can use the method `setTheme` of the desktop (Scout Classic and Scout JS). The chosen theme will be stored in a cookie and activated again on the next visit.

New Servlet Filters to Create a Scout RunContext

Before Scout 9 the `ServerRunContextFilter` was used to create Scout server contexts for REST APIs. This filter used a user based TTL cache that was not bound to the HTTP session.

Starting with Scout 9 there are two new filters available:

- `HttpRunContextFilter`: Creates a Scout run-context without HTTP- and server sessions for stateless REST backends. It supports subject, correlationId, locale, transaction, etc.
- `HttpServerRunContextFilter`: Creates a Scout server-run-context that additionally has a user-agent and an optional Scout server session.

New Widgets

Mode Selector

The widget *ModeSelector* was added. It has similar functionality as the *RadioButtonGroup* but with another design.

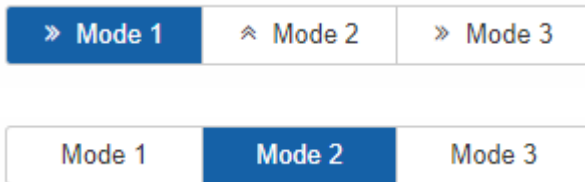


Figure 2. Mode Selector

Popup

It is actually not really a new widget, since it has been used by Scout itself for some other widgets like *SmartField*, *DateField* or *ContextMenu*. What's new on this release is that you can use it as Scout developer, for Scout JS as well as Scout Classic. The *Popup* has the following features:

- Take any widget you like and open it in a *Popup* by using the *WidgetPopup*.
- Use any widget you like as anchor and align the *Popup* around it.
- Decide whether you want to point the *Popup* to the anchor by using the property *withArrow*.
- Control the behavior of what should happen if there is not enough space to display the whole *Popup* using various properties.
- Choose how the popup should react when the user clicks on the outside or on the anchor.

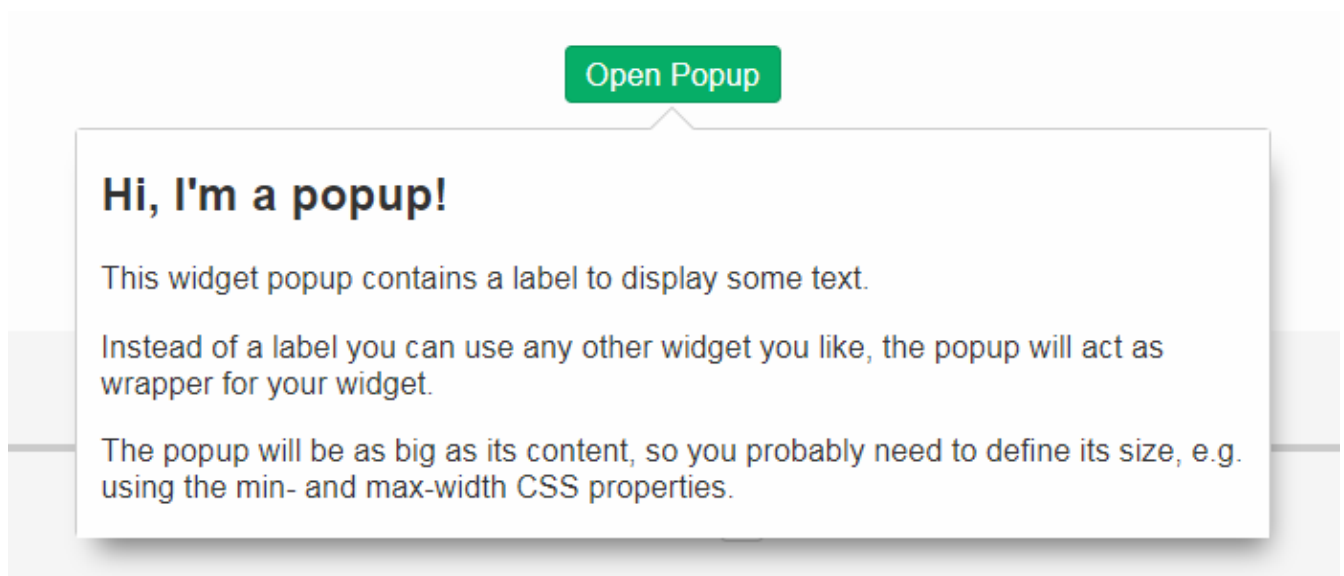


Figure 3. Popup

Check out these features and more in the widget apps!

Label

The new `Label` is a very basic widget to display text. The difference to the existing `LabelField` is that it doesn't extend the `FormField` but the `Widget`. This has the advantage that it is easier to use without the overhead of the `FormField` meaning it is more lightweight. But you cannot use it on regular forms since a form only accepts form fields.

Disabling Close- & Cancel-Buttons

Until Scout 8.0 a Close- or Cancel-Button ignored the enabled-granted property to ensure the form can be closed even if the full form has been disabled using `setEnabledGranted(false)`. This was confusing because the same convenience was not available for all other enabled dimensions.

Since Scout 9.0 Close- and Cancel-Buttons can be disabled like any other form field. But one special handling is still present: The method `isEnabledIncludingParents` ignores the enabled state of the parents and always returns the state of the button only.

So if a Form or GroupBox is disabled using `setEnabled(false)` or `setEnabledGranted(false)` or any other dimension, the full form gets disabled except the Close- and Cancel-Buttons. As soon as the button is disabled explicitly (e.g. by calling `setEnabled(false)` on the button itself or by propagating to the button using `setEnabled(false, false, true)` on a parent composite) it will be disabled and the form cannot be closed anymore.

Improved Scrollbar Usability

The layout structure of the scrollbar comes now with an additional div, and the positioning of the scrollbar uses now padding instead of margin.

With this change, the usability of the scout scrollbar has improved. The thumb is now easier to catch, especially when positioned at the very edge of the screen.

Design Change for WizardProgressField

The wizard progress has a new design.



Figure 4. Wizard Progress

Wizard steps can now be marked as finished, in this case they will be displayed with a check mark icon in the wizard progress.

Improvements for Pages in Scout JS Applications

The API to work with Pages (`PageWithTable`, `PageWithNodes`) has been improved. It is now possible to declare child pages in the static JSON model of outlines and the table within a `PageWithTable` has a default reload handler installed.

Now the method `_loadTableData` (which is responsible for fetching data for a `PageWithTable`) also gets an optional argument `searchFilter` holding the exported data of the first form that is attached to the table using a `FormTableControl` (typically the `SearchForm`). This makes it easier to use the values from a search form by e.g. passing them to a REST backend to limit the results returned from the server.

Finally the `TreeNode` (and therefore all pages because they are tree nodes) get a method `_jsonModel` to declare the static JSON model that belongs to that tree node or page. This works the same way as with all other widgets now.

New Event "lookupCallDone"

All fields having lookup calls (ListBox, RadioButtonGroup, SmartField, TagField) now fire a new event '**lookupCallDone**' always when a lookup call has been executed and the result was processed by the field.

Property Lookup Order Changed

The Scout properties are now resolved in a slightly different order ([Bug 541099](#)). The environment variables are now resolved *before* the `config.properties` file.

1. System properties
2. Environment variables
3. Config properties file
4. Default value of property

Using environment variables, it is now possible to override values in the configuration file, as is already possible using system properties (`-D` flags on JVM command line). This change should simplify the usage of Scout in environments where the application should be static (example: Kubernetes, Docker), but still allow a degree of flexibility.

Since environment variables are not allowed to contain dots/periods (`.`), the new lookup also searches for an equivalent environment variable by replacing periods with underscores (`_`) and converting the property to uppercase.

New CheckableStyle for Table and Tree

For both Table and Tree a new CheckableStyle was added. With the CHECKBOX_TABLE_ROW/CHECKBOX_TREE_NODE style it's possible to check/uncheck a row or node by clicking basically anywhere on the row or node. This new CheckableStyle is now the default in AbstractTree and AbstractListBox. With this CheckableStyle active, expansion on double click is not supported for enabled rows/nodes, since it interferes with the checking/unchecking action.

Strings Sorted with "Natural" Collator by Default

Scout now enables the `NaturalCollatorProvider` by default. When comparing text using a collator (e.g. via `StringUtility`), strings are now sorted more "naturally". Unlike with the JVM default, spaces (" ") and hyphens ("-") are no longer ignored.

This is an old [bug fix](#) that was finally made permanent.

Example:

Listing 1. Input list (unordered)

```
[ "The dogs bark", "The dog barks", "The dog sleeps" ]
```

Listing 2. Sorted list with JVM default (< Scout 9)

```
The dog barks
The dogs bark
The dog sleeps
```

Listing 3. Sorted list with NaturalCollatorProvider (⇒ Scout 9)

```
The dog barks
The dog sleeps
The dogs bark
```

Projects that wish to keep the existing behavior can do so by providing their own `CollatorProvider` (see migration guide).

New Properties for MenuBar Design

There are several new properties added to adapt the design of the MenuBar.

New GroupBox Property 'menuBarPosition'

GroupBoxes can now define the position of the MenuBar inside the GroupBox, the three possibilities are:

- `MENU_BAR_POSITION_AUTO`
- `MENU_BAR_POSITION_TOP`
- `MENU_BAR_POSITION_BOTTOM`

The default value is `MENU_BAR_POSITION_AUTO`, which corresponds to the old behavior.

New GroupBox Property 'menuBarEllipsisPosition'

GroupBoxes can define the position of the ellipsis dropdown menu inside the MenuBar. The possible values are:

- `MENU_BAR_ELLIPSIS_POSITION_LEFT`
- `MENU_BAR_ELLIPSIS_POSITION_RIGHT`

The default value is `MENU_BAR_ELLIPSIS_POSITION_RIGHT`, as it was in earlier releases.

New Menu/Button Property 'shrinkable'



Menus and buttons can define if they are shrinkable or not. When there is not enough space for all menus/buttons in the MenuBar, only the configured Icon of the shrinkable menu/button will be displayed, without text/label. By default the menus/buttons are not shrinkable.

New Button Property 'stackable'


Now not only menus but also buttons can define if they are stackable or not. When after shrinking there is still not enough space in the MenuBar to display all menus/buttons, the stackable menus/buttons will be stacked in the ellipsis dropdown menu. By default the menus/buttons are stackable.

When the ellipsis position inside the MenuBar is `MENU_BAR_ELLIPSIS_POSITION_RIGHT`, the ellipsis menu is placed after the last visible, stackable menu/button. When the ellipsis position is `MENU_BAR_ELLIPSIS_POSITION_LEFT`, the ellipsis menu is placed before the first visible, stackable menu/button.

Group Box

Stackable Button	 Shrinkable Menu	 Shrinkable, not Stackable Menu	Not Stackable Button
String Field 1	<input type="text"/>	String Field 3	<input type="text"/>
String Field 2	<input type="text"/>	String Field 4	<input type="text"/>

Group Box

Stackable Button			Not Stackable Button
String Field 1	<input type="text"/>	String Field 3	<input type="text"/>
String Field 2	<input type="text"/>	String Field 4	<input type="text"/>

Group Box


		Not Stackable Button
String Field 1	<input type="text"/>	
String Field 2	<input type="text"/>	
String Field 3	<input type="text"/>	
String Field 4	<input type="text"/>	

Figure 5. MenuBar layout properties

New OpenUriAction

The URI Action `OpenUriAction.POPUP_WINDOW` is added. The existing URI Action `NEW_WINDOW` leaves it to the browser whether a new tab or a new window is opened. Using the new URI Action `POPUP_WINDOW`, the URI will always be opened in a new window.

New Column Property

'nodeColumnCandidate'

The new property defines if the column can be considered as a candidate for the node column. The node column is used to display the control to expand and collapse rows in a hierarchical table. If **false** the column will be skipped when scanning for the node column and the next suitable column will be chosen as node column.



Do you want to improve this document? Have a look at the [sources](#) on GitHub.