Eclipse Scout

Release Notes

Scout Team

Version 9.0

Table of Contents

About This Release

The Eclipse Scout 9.0 version was released as part of the Eclipse 2019-03 Simultaneous Release ([release schedule](#)) on March 20, 2019.

The latest version of this release is: 9.0.0.018_Simrel_2019_06

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

Service Releases

Scout 9.0 will continue to be maintained for a while and a new build may be released from time to time, following the [Simultaneous Release Cycle](#). Beside bugfixes, these service releases may even contain some minor features.

The following enhancements were made after the initial 9.0 release.

Simrel 2019-06 — Released in June 2019

- [Script File Watcher Enabled by Default (since Simrel 2019-06)](#)

Obtaining the Latest Version

Runtime (Scout RT)

Scout RT artifacts are distributed via Maven:

- [9.0.0.018_Simrel_2019_06](#) on *Maven Central*
- [9.0.0.018_Simrel_2019_06](#) 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>9.0.0.018_Simrel_2019_06</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 Simrel 2019-03](#)

To install the Scout SDK into your existing Eclipse IDE, use this update site:
[http://download.eclipse.org/scout/releases/9.0/9.0.0/018_Simrel_2019_06/](#)

Demo Applications

The demo applications for this version can be found on the features/version/9.0.0.018_Simrel_2019_06 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 11 Support

So far Scout only supported Java 8. In addition to the Java 8 support Scout 9 officially supports OpenJDK 11 too!

Please note that Java 9 and 10 are not supported and that Oracle only provides free Java 8 updates for commercial use until end of January 2019. Therefore it is recommended to use OpenJDK 11.0.1 or newer. Please refer to the Migration guide for instructions how to migrate.

In the future Scout plans to support each LTS (long term support) version of Java. After Java 11 the next supported version might by Java 17. Please note that these plans may change at a future release.

However there are some limitations when using Java 11:

- For Red Hat based Linux Distributions: Red Hat builds its own JDK with some modifications. One modification is that it contains a drastically reduced set of Elliptic Curves (3 instead of the 60+ that OpenJDK contains). Unfortunately the one that Scout uses by default is not part of the Red Hat selection. For more details see 1, 2, 3 and 4. Therefore Red Hat JDK builds are not supported out of the box. However Scout can be configured to work with such JDKs by using a curve that is available in a Red Hat JDK. This can be done by replacing the bean `org.eclipse.scout.rt.platform.security.SunSecurityProvider` and overwriting the method `getEllipticCurveName()`. You can choose a curve that is available on your Red Hat installation. Please note that you need to generate a new key pair for your modified curve using `org.eclipse.scout.rt.platform.security.SecurityUtility.main(String[])` and insert the keys in your config.properties files. Alternatively it is also possible to download an unmodified JDK from the official download page.

- If you are using the SVG support of Scout: There might by compile errors stating that `org.w3c.dom.Element` or similar cannot be resolved. This is a bug in the current Batik version. Batik is the framework Scout uses for SVG support. For more details and how to address it please see Bug 543573.

New SDK Feature

Add a menu entry `Scout` → `Search for duplicate @ClassId Values` to manually trigger a search for duplicate class ids in the workspace.

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.

Responsiveness

Scout group boxes will react reduced widths by transforming its contents to allow better readability. There are three responsive states. The default state is the 'normal' state. In this state no transformations will be applied. In the 'condensed' state, all the label positions will be set to 'TOP'. In the 'compact' state, a one column layout will be enforced. This last state is only available in a Scout-JS application.
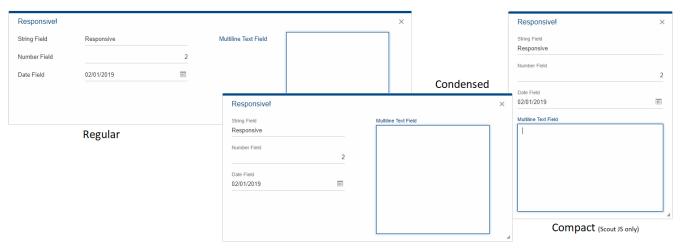

Figure 2. Responsive Form

Once a group box becomes smaller than its preferred width it will transform its internal fields to increase readability. For more information consult the technical guide.

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.

Script File Watcher

During development, the JavaScript and CSS files are always rebuilt when requested by the browser. Depending on the size of the JS and Less code this may take a while, especially the processing of the Less code is expensive. This is unfortunate if only JavaScript code is adjusted or even if no JS or CSS code is touched at all.

To improve this, a Script File Watcher can now be activated which watches all the JS and Less files and triggers a rebuild only if a file changes. This means the files are not always rebuilt on every page reload anymore but only if a relevant file changes. In order to activate the watcher, you need to set the property `scout.dev.scriptfile.rebuild` to false.

Script File Watcher Enabled by Default (since Simrel 2019-06)

The watcher is now enabled by default when running in dev mode. This means, you don't have to set the property `scout.dev.scriptfile.rebuild` anymore, unless you want to disable the watcher.

New Widgets

Mode Selector

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

Figure 3. 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 4. 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 scollbar 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 5. 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.
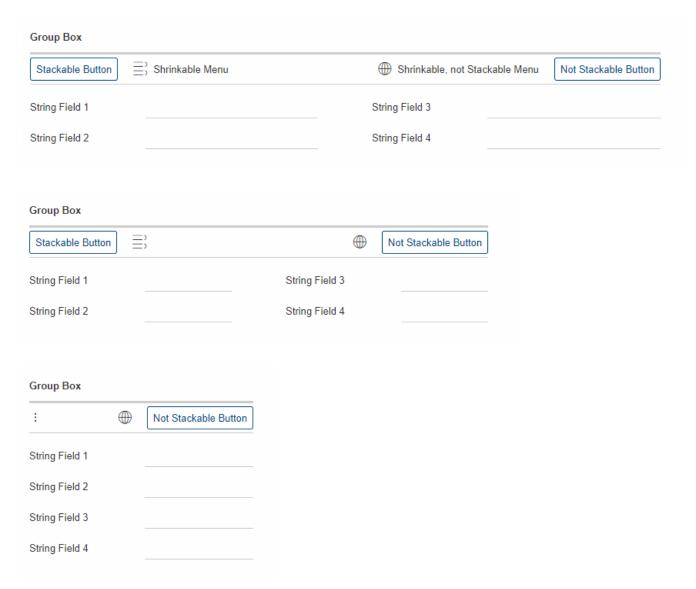
Group Box

| Stackable Button | ☰ Shrinkable Menu | ⊕ Shrinkable, not Stackable Menu | Not Stackable Button |

String Field 1 _____ String Field 3 _____

String Field 2 _____ String Field 4 _____

Group Box

| Stackable Button | ☰ | ⊕ | Not Stackable Button |

String Field 1 _____ String Field 3 _____

String Field 2 _____ String Field 4 _____

Group Box

| ⋮ | ⊕ | Not Stackable Button |

String Field 1 _____

String Field 2 _____

String Field 3 _____

String Field 4 _____

Figure 6. 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.

MOM: Add Support for Handling Incoming/Outgoing JMS Messages

The new interface `org.eclipse.scout.rt.mom.jms.IJmsMessageHandler` adds suport for incoming and outgoing JMS message handling. The default implementation `org.eclipse.scout.rt.mom.jms.LogJmsMessageHandler` logs all messages on level DEBUG.

PropertyChange Event on HtmlEnvironment

Now the HtmlEnvironment fires a `propertyChange` Event when it gets (re)initialized. All Layouts or other Components that depend on the HtmlEnvironment's properties should listen to that event and handle it appropriately.

Dense Mode

This new display mode is targeted at users working on laptops or other small screens. It reduces the whitespace between elements to display more content on the available screen size.

Activate the dense mode by setting `IDesktop.setDense(true)`. An additional less file `displaystyle-dense.less` contains all necessary styling changes.
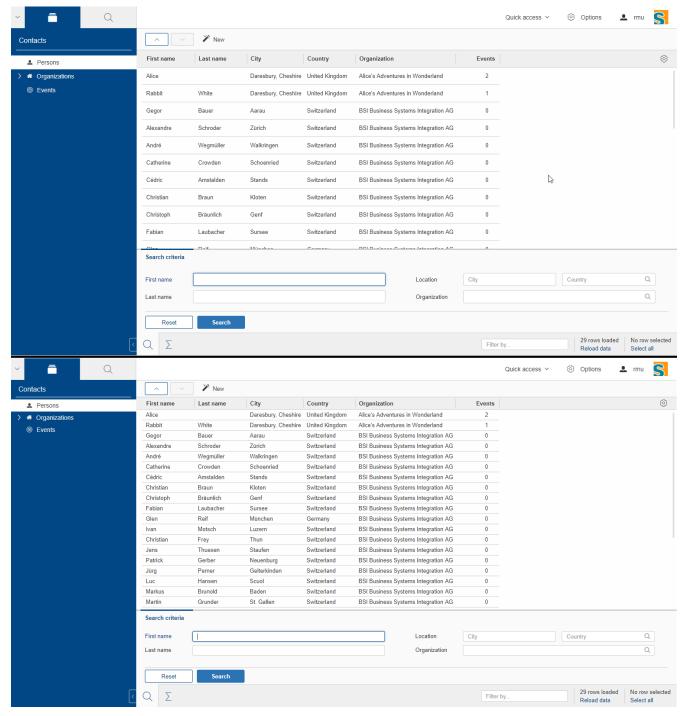

Figure 7. Dense Mode

Copy to clipboard support for MessageBoxes

The `MessageBox` widget now supports the OS specific copy to clipboard key-shortcut. When pressed, all text of the MessageBox, including the *hiddenText* property, is copied to the clipboard.

Removed Dependency to java.util.ResourceBundle

As `NlsResourceBundle` does not use any built features of `java.util.ResourceBundle`, it does no longer depend on `java.util.ResourceBundle`.

Configuration property `scout.resourceBundle.checkContainsKey` was removed as it is now obsolete.

Improvements for 'Group' widget

Widget in header

It is now possible to define a custom widget as group header instead of only the title/suffix texts. This allows for example to add input-fields or buttons to group headers. This can be achieved by overriding the method `getConfiguredHeader` and returning a widget class or `createHeader` and returning the widget instance.

Collapse style 'bottom'

The property `collapseStyle` was extended by the style `bottom` which displays a bottom border for the group and (if the group is collapsible) a collapse-arrow centered in the middle of the bottom-border.

Property 'visible'

The property `visible` in the Java model was migrated from a `boolean` to a `byte` type to achieve multiple dimension support for the visibility of a group (see also https://eclipsescout.github.io/9.0/technical-guide.html#multiple-dimensions-support).

Data Objects: Equals and HashCode (since 8.0.0.036)

The Data Object base class `DoEntity` was extended with a generic `equals()` and `hashCode()` implementation considering all attributes of a data object for equality. A data object is equals to another data object, if the Java class of both data objects is identical and the attribute maps (including their nested values) of both data objects are equals.

For futher details see:

- `org.eclipse.scout.rt.platform.dataobject.DoEntity.equals(Object)`
- `org.eclipse.scout.rt.platform.dataobject.DoNode.equals(Object)`

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