Eclipse Scout

Release Notes

Scout Team

Version 6.1

Table of Contents

Releases

Oxygen: This version is under development and scheduled for release in June 2017.

- Download SDK: Eclipse for Scout Developers
- Runtime on Maven Central: 6.1.0.x

What's New

Strong content security policy CSP

The stronger CSP disables inline javscript in html. Therefore the 'New Scout Project' wizard now creates a js file per html file and includes it using the script element. To migrate existing projects, see the Scout Migration Guide.

Enforcement of Model Thread

Every operation which results in a modification of the model and eventually of the ui has to be performed by the model thread. This has been true for a long time and still is. If the wrong thread was used, unexpected behavior was the result, like a delayed update of the ui or concurrency exceptions. To prevent such behavior in the future, an exception will be thrown if an operation is executed in the wrong thread.

If you get such an exception, you'll need to wrap your operation in a model job and schedule it using `ModelJobs.schedule()`, see the chapter ModelJobs in the Scout Technical Documentation for details.

Binary resources support for HtmlField and BeanField

Binary resources such as images or videos can now be used in the following widgets:

- HtmlField
- BeanField

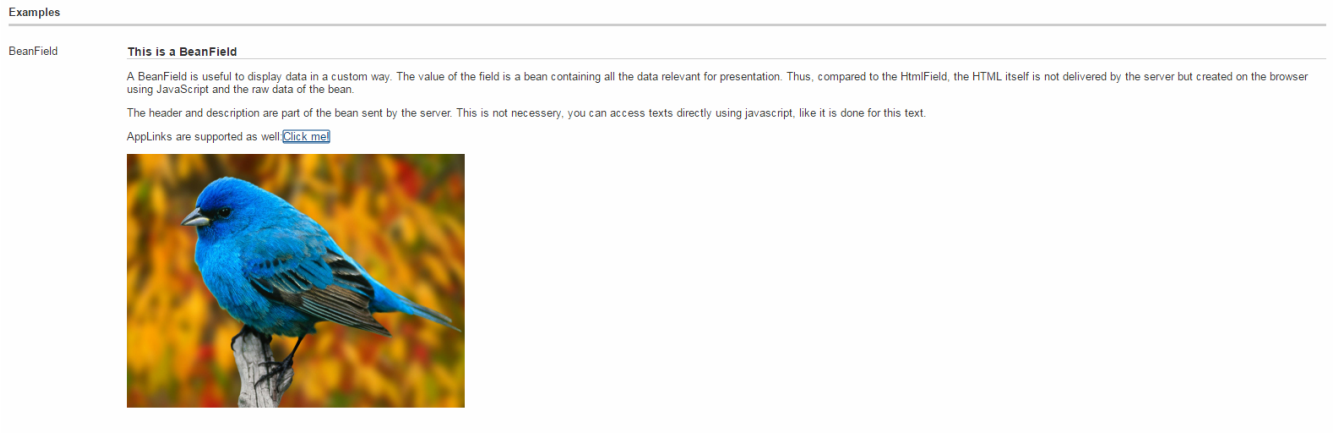

Figure 1. Binary resource on a model field.

- Html enabled StringColumn
- BeanColumn

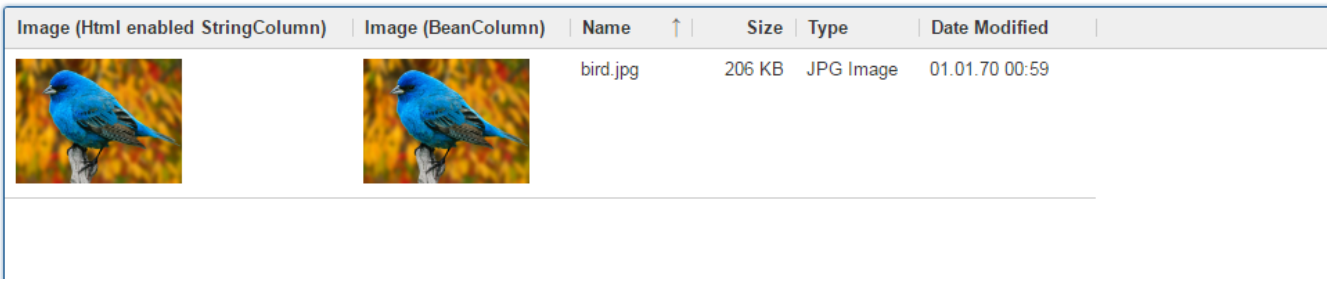

Figure 2. Binary resource on a column

TriState capability for check boxes

Added support for tri-state value (`true`, `false` and `null` instead of just `true` and `false`) to boolean field and boolean column.

The new property `triStateEnabled` controls whether the boolean field/column behaves as a normal checkbox (false) or a tri-state checkbox (true).

A normal checkbox has values true/false. A tri-state checkbox has values true/false/null. The null value is interpreted as "undefined" and rendered as a filled square.

New UriOpenAction: SAME_WINDOW

The enum `UriOpenAction` provides a new value to open a URI in the current window: `SAME_WINDOW`

New methods in StringUtility

`StringUtility` provides the following new methods:

- `containsString()`
- `containsStringIgnoreCase()`
- `containsRegEx()`

- `matches()`
- `endsWidth()`
- `startsWith()`
- `length()`
- `indexOf()`
- `lastIndexOf()`
- `split()` (with *limit* argument)

All methods are null-safe, unit tested and documented with JavaDoc.

New ObjectUtility

ObjectUtility was added as new utility for generic object methods and provides null-safe implementations of various Object methods. Various methods from former `CompareUtility`:

- `equals()`
- `notEquals()`
- `nvl()`
- `isOneOf()`
- `compareTo()`

And a new method `ObjectUtility.toString(Object)` providing a null-safe implementation of `Object.toString()` returning `null` if specified object is `null`.

New form field property "preventInitialFocus"

By default, the first enabled field on a form gets the focus when the form is opened. This may not be desired in some cases (e.g. if the first field is a HTML field that contains app links). The new property `PROP_PREVENT_INITIAL_FOCUS` can be used to prevent the initial focus to be set to this field. The default value is `false`. For `AbstractHtmlField` and `AbstractBeanField`, the default is set to `true`.

Multiple Dimensions Support

Some components now support more dimensions for various attributes. E.g. until now there have been two dimensions for Form Field visibility: visible and visible-granted. Now there are also custom dimensions available. See the chapter 'Multiple Dimensions Support' in the Scout Technical Documentation for details and examples.

Currently the following attributes support multiple dimensions:

- Actions: visible, enabled
- Columns: visible
- Tree Nodes: visible, enabled

- Outlines: visible
- Form Fields: visible, enabled, label-visible
- Data Model Attributes: visible
- Data Model Entity: visible
- Wizard Steps: visible, enabled
- Trees: enabled
- Tables: enabled

Form Field Enabled Inheritance

The inheritance of the enabled property for Form Fields has been changed. Now the enabled properties are no longer propagated to children if it is changed on a composite field. Instead a field is only considered to be enabled if itself and all of its parents are enabled. This allows to toggle an entire box to disabled and back to enabled without touching the child fields. This has the advantage that the original state is restored when the box is set back to enabled.

With this change the `getConfiguredEnabled` on composite fields now also automatically affects children. There is no need to overwrite `execInit()` and call `setEnabled(false)` anymore.

Do you want to improve this document? Please edit this page on GitHub.