Eclipse Scout
Migration Guide

Scout Team

Version 8.0

Table of Contents

About This Document

This document describes all relevant changes **from Eclipse Scout 7.0 to Eclipse Scout 8.0**. If existing code has to be migrated, instructions are provided here.

Service Release Migration

The following changes were made after the initial 8.0 release (Eclipse Photon release). Additionally follow these instructions when updating to a *service release*.

Photon.1 (8.0.100) Release Expected on September, 2018

- SmartField: changed behavior for applying styles from a lookup-row on the field (since 8.0.100)
- Moved mail related classes to new module to org.eclipse.scout.rt.mail (since 8.0.100)
- Call: Skip Retries by Default (since 8.0.100)
- ErrorHandler: Extended Support for Arbitrary Errors (since 8.0.100)
- SmartField / DateField: Renamed touch to touchMode (since 8.0.100)

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

API Changes (Java)

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 yet. The Java 9 support is planned for Eclipse *Photon* release (Scout 8.0) in summer 2018.

To reflect this change, some existing Scout classes have been migrated to the newly available Java 8 classes:

Old Scout class:	Replaced with Java 8 equivalent:
`org.eclipse.scout.rt.platform.filter.IFilter`	`java.util.function.Predicate`
`org.eclipse.scout.rt.platform.filter.AlwaysFilter`	`java.util.function.Predicate` e.g. `a → true`
`org.eclipse.scout.rt.platform.filter.NotFilter`	`java.util.function.Predicate#negate`
`org.eclipse.scout.rt.platform.filter.OrFilter`	`java.util.function.Predicate#or`
`org.eclipse.scout.rt.platform.util.concurrent.IConsumer`	`java.util.function.Consumer`
`org.eclipse.scout.rt.platform.util.concurrent.IFunction`	`java.util.function.Function`

Old Scout class:	Replaced with Java 8 equivalent:
`org.eclipse.scout.rt.platform.util.concurrent.IBiConsumer`	`java.util.function.BiConsumer`
`org.eclipse.scout.rt.platform.util.concurrent.IBiFunction`	`java.util.function.BiFunction`
`org.eclipse.scout.rt.client.ui.action.IActionFilter`	`java.util.function.Predicate<IAction>`

CalendarComponent

Removed deprecated method `Date[] getCoveredDays()` on `org.eclipse.scout.rt.client.ui.basic.calendar.CalendarComponent`.

Migration: Use `Range<Date> getCoveredDaysRange()` instead with start and end date.

AbstractCacheValueResolver

Abstract implementation was replaced by default method in interface `ICacheValueResolver`.

Migration: Implement interface `ICacheValueResolver` directly instead of extending `AbstractCacheValueResolver`.

Deprecation of SequenceBox EqualColumnWidths

The methods `SequenceBox#getConfiguredEqualColumnWidths()`, `SequenceBox#isEqualColumnWidths()` and `SequenceBox#setEqualColumnWidths()` have been deprecated. As this property was not used by Scout since several releases it can be deleted.

You can search for these method names in your code to find any usages. Alternatively your IDE may mark the usage of deprecated methods (depends on settings).

Removed Constants/Methods From ClientUIPreferences

Since the new Html UI was introduced with Scout 5.2 some constants and methods in `ClientUIPreferences` had no effect on the UI anymore. Since Scout is now a browser application we don't have any control over the application window anymore. The form bounds are stored in the local browser storage thus they're not required in the Java model anymore.

The following constants/methods were dropped:

- ~~`APPLICATION_WINDOW_MAXIMIZED`~~
- ~~`APPLICATION_WINDOW_BOUNDS`~~
- ~~`FORM_BOUNDS`~~
- ~~`getFormBounds()`~~
- ~~`setFormBounds(IForm form, Rectangle bounds)`~~
- ~~`getApplicationWindowMaximized()`~~

- ~~setApplicationWindowPreferences(BoundsSpec r, boolean maximized)~~

Removed Property "focusable" From IFormField, AbstractFormField

Since the new Html UI was introduced with Scout 5.2 the property `focusable` had no effect on the UI anymore. Instead the UI uses sensible defaults for each field type. For instance: a *LabelField* is never focusable, a normal *StringField* is always focusable, as long as it is enabled. Since the property was rarely used, we removed the related code:

- Method ~~boolean IFormField#isFocusable()~~
- Method ~~void IFormField#setFocusable(boolean f)~~
- Field ~~boolean IFormField#PROP_FOCUSABLE~~
- Method ~~boolean AbstractFormField#getConfiguredFocusable()~~
- Class ~~AbstractNonFocusableButton~~
- Class ~~AbstractNonFocusableRadioButton~~

Migration: Since there is no replacement for the `focusable` property, remove all code that uses one of the methods/properties listed above.

Removal of Old "SmartField", Renaming of "SmartField2" Back to "SmartField"

In Scout 7.0 a new smart field implementation named *SmartField2* was created. The old implementation was still available, but its use was discouraged. With Scout 8.0 the old *SmartField* implementation was finally dropped. The new implementation *SmartField2* was renamed back to *SmartField*.

The following table lists the different names used in the past Scout releases. (The naming applies to all associated files, such as interfaces, abstract field classes, Java packages and JavaScript and LESS files.)

Scout <= 6.1	**Scout 7.0**	**Scout >= 8.0**
SmartField *(old implementation)*	SmartField *(old implementation)*	—
—	SmartField2 (new implementation)	SmartField (new implementation)

Migration:

- If you have already used the new *SmartField2* in Scout 7.0 you must move / rename all references back to the (new) *SmartField*.
- If you are migrating from an older Scout version (<= 6.1), you *might* have to adjust your code to the new implementation. See the corresponding 7.0 migration guide.

IContentAssistField Classes and Interfaces Removed

The base class and interfaces of the old SmartField implementation was IContentAssistField. With the new SmartField implementation this interfaces and all classes containing the word "ContentAssist" in their name, have been either removed or renamed to "SmartField". If your code references one of these classes you should simply try to rename all references. Since the API of the new SmartField is almost the same as the old, this should work and should cause no or few changes in your code. The following classes have been removed without replacement:

- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.AbstractMixedSmartField`~~
- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.ContentAssistFieldEvent`~~
- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.ContentAssistFieldListener`~~
- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.ContentAssistFieldTable`~~
- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.IMixedSmartField`~~

Since the new SmartField implementation does not have a proposal chooser model anymore these classes have also been removed. If you must have a special implementation of a proposal chooser, you must implement a proposal chooser in JavaScript (see: *ProposalChooser.js*), which renders the data and lookup rows it receives from the server-side SmartField. The following classes have been removed without replacement:

- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.AbstractProposalChooser`~~
- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.IProposalChooser`~~
- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.IProposalChooserProvider`~~
- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.TableProposalChooser`~~
- ~~`org.eclipse.scout.rt.client.ui.form.fields.smartfield.TreeProposalChooser`~~

GroupBox: Moved "minWidthInPixel"

The property `minWidthInPixel` has been moved to *LogicalGridLayoutConfig*.

Migration: Instead of using `getConfiguredMinWidthInPixel` you should now set the property as follows:

```
@Override
protected LogicalGridLayoutConfig getConfiguredBodyLayoutConfig() {
  return super.getConfiguredBodyLayoutConfig()
      .withMinWidth(400);
}
```

Adjusted Behavior of Widget Initialization

The goal was to harmonize all the init methods (`initField`, `initTile`, `initForm` etc.) and to make sure, `init()` is only executed once. This is important for dynamic widgets like accordion or tiles. These widgets initialize the newly added children by themselves so that the caller does not need to take care of it. For these cases it is important that `init()` is not called multiple times.

But: there may be cases which require `init()` to be called multiple times, like reseting a search form. For such cases, `reinit()` has to be used from now on. Also, after the widget is disposed, `init()` may be called again. So remember: `execInit` may be called more than once in some circumstances. This is existing behavior!

We also renamed the initConfig guard of `IFormField` from `isInitialized` to `isInitDone` to make clear what initialization has been done. It has furthermore been moved to `AbstractWidget` so that individual widgets don't have to care about it and to use the same pattern as for `init` and `dispose`.

These new methods (`init()`, `dispose()` and `reinit()`) handle the whole widget tree including all child widgets recursively. Child items that need initialization and are NO widgets must be initialized explicitly as it is already now. All children that are widgets must NOT be initialized because this is done automatically by the AbstractWidget implementation. To modify that behavior use the methods `initChildren()` or `disposeChildren()` (e.g. if you want to exclude a child widget from automatically getting initialized). This also means that the methods `ActionUtility.initActions()`, `ActionUtility.disposeActions()`, `FormUtility.postInitConfig()`, `FormUtility.initFormFields()` and `FormUtility.disposeFormFields()` have been removed because the corresponding method can be called directly on the instance instead. Use `widget.init()` or `widget.dispose()` instead of these utility functions.

Please note that the phase `postInit` has been removed for the items that supported it. The corresponding code can be moved to the end of the `initConfigInternal()` method instead.

We also renamed the initConfig guard of `IFormField` from `isInitialized` to `isInitDone` to make clear what initialization has been done. It has furthermore been moved to `AbstractWidget` so that individual widgets don't have to care about it and to use the same pattern as for `init` and `dispose`.

Migration:

If you used one of the deprecated methods (`initField`, `initAction` etc.), replace them with one of the following methods: `init`, `reinit` or `initInternal`.

- Use `init` if you created a field and need to initialize it.
- Use `reinit` if you explicitly want to reinitialize an already initialized field.
- Use `initInternal` if your custom widget overrides `initField`.

GzipServletFilter

Replaced init parameters `get_pattern` and `post_pattern` with `content_types`. If you set these init parameters in your web.xml, replace or remove them accordingly.

HttpCacheControl

The argument `pathInfo` has been removed from the method `HttpCacheControl.checkAndSetCacheHeaders` since it has no effect anymore.

HttpProxy

HTTP Proxy doesn't set cache control `no-chache` header anymore.

Replacement for IActionVisitor, ITreeVisitor, IFormFieldVisitor

A new tree visitor engine has been added to the `org.eclipse.scout.rt.platform.util.visitor` package. It contains classes to depth-first or breadth-first traverse any tree data structures. Use the class `org.eclipse.scout.rt.platform.util.visitor.TreeTraversals` as entry point.

This new visitors can be used on any widget and tree node. It replaces the former IActionVisitor, ITreeVisitor and IFormFieldVisitor. The `org.eclipse.scout.rt.client.ui.IWidget` interface also declares various overloads accepting `java.util.function.Consumer`, `java.util.function.Function` and the new `org.eclipse.scout.rt.platform.util.visitor.IDepthFirstTreeVisitor` and `org.eclipse.scout.rt.platform.util.visitor.IBreadthFirstTreeVisitor`. Depending on how much control and data you need for your visitor the matching type can be used.

Until now only pre-order visitors have been available on these items. Therefore the migration to the new visitor takes the following steps:

- `visitFields` or `acceptVisitor` method on IFormField or `IAction` have been replaced with the `visit` method on IWidget.
- The new visitor allows to control how visiting should be continued in a more detailed level. Instead returning `true` or `false` to indicate if visiting should continue all options as defined in `org.eclipse.scout.rt.platform.util.visitor.TreeVisitResult` are available. A return value of `TreeVisitResult.CONTINUE` corresponds to `true` and a return value of `false` can be migrated to `TreeVisitResult.TERMINATE`.
- If using the `IDepthFirstTreeVisitor` the method `preVisit` must be overridden to have the same functionality as before. Consider also using the `DepthFirstTreeVisitorAdapter` instead of directly using the interface.
- If the visitor should only be called for a certain type of input element and just continue visiting for all others the overloads defining a type filter can be used for widgets. Using this instanceof checks and type casts are often not necessary anymore.
- There is a `org.eclipse.scout.rt.platform.util.visitor.CollectingVisitor` class to convert the items of a tree to a list.
- IFormField.visitParents takes a `java.util.function.Predicate<IFormField>` instead of an IFormFieldVisitor.

New Convenience Methods on IFormField

There are new methods for setting mandatory state (`setMandatory`), status visibility (`setStatusVisible`), field style (`setFieldStyle`) and disabled style (`setDisabledStyle`) that allow to specify if child form fields should be changed as well.

So if you have overridden one of these methods in your code, please override the new one instead. The method now takes an additional boolean flag to indicate if children should be processed as well.

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.

All properties must now implement this new method and return a description text of that property. The class `org.eclipse.scout.rt.platform.config.ConfigDescriptionExporter` can be used to export these descriptions. By default an AsciiDoctor exporter is included.

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 concreate 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 the platform will still start.

To minimize configuration files such entries should be removed from `config.properties` files.

Renamed Config Property Keys

The following config property keys have been renamed (the old keys are no longer valid and must be renamed accordingly):

Table 1. Config Property Renames

Old Key	New Key
`scout.auth.anonymous.enabled`	`scout.auth.anonymousEnabled`
`scout.auth.cookie.enabled`	`scout.auth.cookieEnabled`
`scout.auth.cookie.maxAge`	`scout.auth.cookieMaxAge`
`scout.auth.cookie.name`	`scout.auth.cookieName`
`scout.auth.cookie.session.validate.secure`	`scout.auth.cookieSessionValidateSecure`
`scout.auth.credentials.plaintext`	`scout.auth.credentialsPlaintext`
`scout.auth.token.ttl`	`scout.auth.tokenTtl`

Old Key	New Key
scout.server.url	scout.backendUrl
session.jobCompletionDelayOnSessionShutdown	scout.client.jobCompletionDelayOnSessionShutdown
org.eclipse.scout.memory	scout.client.memoryPolicy
notification.user.authenticator	scout.client.notificationSubject
org.eclipse.scout.testing.client.ClientSessionProviderWithCache#expiration	scout.client.testingSessionTtl
user.area	scout.client.userArea
org.eclipse.scout.rt.server.clientnotification.ClientNotificationService#maxMessages	scout.clientnotification.chunkSize
org.eclipse.scout.rt.server.clientnotification.ClientNotificationService#blockingTimeout	scout.clientnotification.maxNotificationBlockingTimeOut
org.eclipse.scout.rt.server.clientnotification.ClientNotificationNodeQueue#capacity	scout.clientnotification.nodeQueueCapacity
org.eclipse.scout.rt.server.clientnotification.ClientNotificationRegistry#m_queueExpireTime	scout.clientnotification.notificationQueueExpireTime
org.eclipse.scout.rt.server.services.common.clustersync.ClusterSynchronizationService#user	scout.clustersync.user
scout.beans.createTunnelToServerBeans	scout.createTunnelToServerBeans
scout.csp.enabled	scout.cspEnabled
scout.csp.directive	scout.cspDirective
scout.dev.mode	scout.devMode
scout.external.base.url	scout.externalBaseUrl
scout.healthcheck.remoteUrls	scout.healthCheckRemoteUrls
scout.http.apache_connection_time_to_live	scout.http.connectionTtl
scout.http.ignore_proxy	scout.http.ignoreProxyPatterns
scout.http.apache_keep_alive	scout.http.keepAlive
scout.http.apache_max_connections_per_route	scout.http.maxConnectionsPerRoute
scout.http.apache_max_connections_total	scout.http.maxConnectionsTotal
scout.http.proxy	scout.http.proxyPatterns
scout.http.apache_retry_post	scout.http.retryPost
scout.http.transport_factory	scout.http.transportFactory
org.eclipse.scout.rt.server.services.common.imap.AbstractIMAPService#host	scout.imap.host
org.eclipse.scout.rt.server.services.common.imap.AbstractIMAPService#mailbox	scout.imap.mailbox
org.eclipse.scout.rt.server.services.common.imap.AbstractIMAPService#password	scout.imap.password
org.eclipse.scout.rt.server.services.common.imap.AbstractIMAPService#port	scout.imap.port
org.eclipse.scout.rt.server.services.common.imap.AbstractIMAPService#sslProtocols	scout.imap.sslProtocols
org.eclipse.scout.rt.server.services.common.imap.AbstractIMAPService#userName	scout.imap.username
jandex.rebuild	scout.jandex.rebuild
jaxws.consumer.connectTimeout	scout.jaxws.consumer.connectTimeout

Old Key	New Key
jaxws.consumer.portCache.corePoolSize	scout.jaxws.consumer.portCache.corePoolSize
jaxws.consumer.portCache.enabled	scout.jaxws.consumer.portCache.enabled
jaxws.consumer.portCache.ttl	scout.jaxws.consumer.portCache.ttl
jaxws.consumer.portPool.enabled	scout.jaxws.consumer.portPoolEnabled
jaxws.consumer.readTimeout	scout.jaxws.consumer.readTimeout
jaxws.implementor	scout.jaxws.implementor
jaxws.loghandler.debug	scout.jaxws.loghandlerDebug
jaxws.provider.authentication.basic.realm	scout.jaxws.provider.authentication.basicRealm
jaxws.provider.user.authenticator	scout.jaxws.provider.user.authenticator
jaxws.provider.user.handler	scout.jaxws.provider.user.handler
scout.mom.requestreply.cancellation.topic	scout.mom.requestreply.cancellationTopic
scout.node.id	scout.nodeId
scout.permission.level.check.cache.ttl	scout.permissionLevelCacheTtl
org.eclipse.scout.rt.server.services.common.file.RemoteFileService#rootPath	scout.remotefileRootPath
org.eclipse.scout.rt.server.session.ServerSessionProviderWithCache#expiration	scout.serverSessionTtl
org.eclipse.scout.serviceTunnel.compress	scout.servicetunnel.compress
org.eclipse.scout.rt.servicetunnel.apache_max_connections_per_route	scout.servicetunnel.maxConnectionsPerRoute
org.eclipse.scout.rt.servicetunnel.apache_max_connections_total	scout.servicetunnel.maxConnectionsTotal
org.eclipse.scout.rt.servicetunnel.targetUrl	scout.servicetunnel.targetUrl
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#debugReceiverEmail	scout.smtp.debugReceiverEmail
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#defaultFromEmail	scout.smtp.defaultFromEmail
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#host	scout.smtp.host
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#password	scout.smtp.password
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#port	scout.smtp.port
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#sslProtocols	scout.smtp.sslProtocols
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#subjectPrefix	scout.smtp.subjectPrefix
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#useAuthentication	scout.smtp.useAuth
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#username	scout.smtp.username
org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService#useSmtps	scout.smtp.useSsl
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#directJdbcConnection	scout.sql.directJdbcConnection
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jdbcDriverName	scout.sql.jdbc.driverName

Old Key	New Key
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jdbcDriverUnload	scout.sql.jdbc.driverUnload
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jdbcMappingName	scout.sql.jdbc.mappingName
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jdbcPoolConnectionBusyTimeout	scout.sql.jdbc.pool.connectionBusyTimeout
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jdbcPoolConnectionLifetime	scout.sql.jdbc.pool.connectionIdleTimeout
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jdbcPoolSize	scout.sql.jdbc.pool.size
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jdbcProperties	scout.sql.jdbc.properties
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jdbcStatementCacheSize	scout.sql.jdbc.statementCacheSize
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jndiInitialContextFactory	scout.sql.jndi.initialContextFactory
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jndiName	scout.sql.jndi.name
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jndiProviderUrl	scout.sql.jndi.providerUrl
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#jndiUrlPkgPrefixes	scout.sql.jndi.urlPkgPrefixes
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#password	scout.sql.password
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#transactionMemberId	scout.sql.transactionMemberId
org.eclipse.scout.rt.server.services.common.jdbc.AbstractSqlService#username	scout.sql.username
scout.text.providers.show.keys	scout.texts.showKeys
scout.background.polling.interval	scout.ui.backgroundPollingMaxWaitTime
scout.max.user.idle.time	scout.ui.maxUserIdleTime
scout.ui.model.jobs.await.timeout	scout.ui.modelJobTimeout
scout.sessionstore.housekeepingDelay	scout.ui.sessionstore.housekeepingDelay
scout.sessionstore.housekeepingMaxWaitForShutdown	scout.ui.sessionstore.housekeepingMaxWaitForShutdown
scout.sessionStore.maxWaitForAllShutdown	scout.ui.sessionStore.maxWaitForAllShutdown
scout.sessionStore.valueUnboundMaxWaitForWriteLock	scout.ui.sessionStore.valueUnboundMaxWaitForWriteLock

Customizing CSP Rules Via Config Property

The new config property `scout.cspDirective` makes subclassing and replacing the `ContentSecurityPolicy` class obsolete as you can configure all CSP settings with this property now. An example from the Scout Widgets application:

```
@Replace
public class WidgetsContentSecurityPolicy extends ContentSecurityPolicy {

  @Override
  protected void initDirectives() {
    super.initDirectives();
    // Demo app uses external images in html field and custom widgets -> allow it
    withImgSrc("*");
  }
}
```

This class was deleted and replaced by a config property in *config.properties*:

```
# CSP - Demo app uses external images in html field and custom widgets -> allow it
scout.cspDirective[img-src]=*
```

IUiServletRequestHandler

The methods `handleGet` and `handlePost` on `IUiServletRequestHandler` were replaced by the single method `handle`. This new method is called for all HTTP methods.

To retrieve the HTTP method, call `getMethod` on `HttpServletRequest`. When using `AbstractUiServletRequestHandler` no migration should be required because `AbstractUiServletRequestHandler` delegates to the Java methods for the common HTTP methods `handleGet`, `handlePost`, `handlePut` and `handleDelete`.

Methods `proxyGet` and `proxyPost` on `HttpProxy` are replaced by the common method `proxy`.

Table: Remove Obsolete "POPULATED" Event and Methods

We removed remnants of the long-obsolete "population" event in tables:

- ~~`ITable.tablePopulated()`~~
- ~~`TableEvent.TYPE_TABLE_POPULATED`~~

Migration: Remove any references to the removed method or event from your code. (This should not cause any change in behavior, as the event was not fired by Scout anyways.)

PageField: Status of Page Table Visible by Default

The table field contained in a *PageField* used to have `statusVisible` set to *false*. This default was changed back to *true* to make it consistent with all other fields. Whether the status should be invisible can only be determined correctly by the programmer, because the PageField can not know about the status visibility in the inner forms (search form and search form).

Migration: To hide the status of a specific *PageField*'s table field, override `execInitField()` and set the desired status visibility:

```
public class MyPageField extends AbstractPageField<MyTablePage> {

  @Override
  protected void execInitField() {
    getTableField().setStatusVisible(false); // <--
  }
}
```

SmartField: changed behavior for applying styles from a lookup-row on the field (since 8.0.100)

The SmartField in Scout ⇐ 6.0 applied the lookup-row properties `foregroundColor`, `backgroundColor`, `font` and `tooltipText` automatically on the field when a lookup-row has been selected. Because that automatic behavior didn't fit every business requirement, we removed it completely. This means you must implement the property-synchronization where required.

Since we prefer to do styling with CSS and LESS, we now copy the `cssClass` property from the lookup-row to the field. In some cases this new feature can be used to migrate from older Scout version.

Check the HowTo section *SmartField: how to apply colors and styles from a lookup-row* in Scout's technical guide to find examples for migration and how to work with the `cssClass` property.

Moved mail related classes to new module to org.eclipse.scout.rt.mail (since 8.0.100)

The following classes were moved from `org.eclipse.scout.rt.shared.mail` to a new module named `org.eclipse.scout.rt.mail` to the package `org.eclipse.scout.rt.mail`:

- BinaryResourceDataSource
- CharsetSafeMimeMessage
- MailAttachment
- MailHelper
- MailMessage
- MailParticipant
- RFCWrapperPart

For migration, replace all imports for `org.eclipse.scout.rt.shared.mail` by imports for `org.eclipse.scout.rt.mail` and add a dependency to the new module `org.eclipse.scout.rt.mail`.

A new helper `org.eclipse.scout.rt.mail.smtp.SmtpHelper` replaces the service

`org.eclipse.scout.rt.server.services.common.smtp.ISMTPService` and it's abstract implementation `org.eclipse.scout.rt.server.services.common.smtp.AbstractSMTPService`. The only remaining property is `SmtpDebugReceiverEmailProperty` (`scout.smtp.debugReceiverEmail`), all others must be set in `org.eclipse.scout.rt.mail.smtp.SmtpServerConfig`.

`SmtpHelper` has no support for subject prefix and default from email anymore. If a subject prefix is required, this must be prepended to the subject before calling `SmtpHelper` by using `MailHelper.addPrefixToSubject`. If a default from email is required, this must be ensured before calling `SmtpHelper` by using `MailHelper.ensureFromAddress`.

API Changes (JavaScript)

Rename of LESS Variables

If you created a custom theme, you might have to adjust some LESS variables.

- Splitted `@group-title-padding-y` into `@group-box-title-padding-top` and `@group-box-title-padding-bottom`
- Splitted `@tree-node-padding` into `@tree-node-padding-y`, `@tree-node-padding-left` and `@tree-node-padding-right`
- Renamed `@group-title-border-width` to `@group-box-title-border-width`
- Renamed `@group-margin-bottom` to `@group-box-body-padding-bottom`
- Renamed `@group-margin-top` to `@group-box-body-padding-top`
- Added `@group-box-title-margin-top`
- Renamed `@tabbox-padding-x` to `@tab-item-padding-x`
- Renamed `@tabbox-focus-arrow-width` to `@tab-item-focus-arrow-width`
- Renamed `@tabbox-border-width` to `@tab-area-border-width`
- Renamed `@compact-outline-node-padding-v` to `@compact-outline-node-padding-y`
- Renamed `@box-margin-v` to `@box-margin-y`
- Renamed `@outline-breadcrumb-node-padding-v` to `@outline-breadcrumb-node-padding-y`
- Renamed `@tile-padding-h` to `@tile-field-padding-x`
- Renamed `@tile-padding-v` to `@tile-field-padding-y`
- Renamed `@planner-header-buttons` to `@planner-header-button-height`
- Renamed `@calendar-header-buttons` to `@calendar-header-button-height`
- Renamed `@logical-grid-height` to `@logical-grid-row-height`
- Renamed `@applink-color` to `@link-color`

Widget.js: New Argument for clone()

The *clone()* function of any widget got an `options` parameter. The options define what properties and events are synchronized between the widget and its clone.

Widget.js: "addChild" & "removeChild" Changed to Private

The methods `addChild()` and `removeChild()` have been renamed to `_addChild()` and `_removeChild()`. This means the methods are considered to have `private` visibility now. Use the methods `setParent()`, `setOwner()` and `destroy()` to connect or disconnect widgets. These methods will add or remove the child widget automatically.

FormField.js: "visit()" Renamed to "visitFields"

The `visit()` method on all FormFields has been renamed to `visitFields()`. This change is necessary to clarify what is visited and to distinguish between the visit methods available on widget level (e.g. `visitChildren()`).

Tree._visitNodes & Tree.visitNodes

The argument order of the method `scout.Tree.visitNodes` have changed from (nodes, func, parentNode) to (func, nodes, parentNode). So the func (the visitor) and the nodes to visit have changed positions.

The arguments of `scout.Tree.prototype._visitNodes` have changed from (nodes, func, parentNode) to (func, parentNode). So the nodes to visit must no longer be specified. Instead always the root nodes of the tree are used. Furthermore the method is public now and has therefore be renamed to `visitNodes()`.

Automatic Preloading of Web Fonts

Scout can now detect the web fonts (*.woff) to preload automatically. It's therefore no longer necessary to list the font names manually in the bootstrap argument of `scout.App`.

Migration:

Remove the `fonts` property from the bootstrap parameter object passed to the *init()* function of your Scout app.

For example, the default *index.js* file generated by the Scout "helloworld" archetype looks like this:

```
$(document).ready(function() {
  var app = new scout.RemoteApp();
  app.init({
    bootstrap: {
      fonts: ['scoutIcons'] // <-- this property is no longer required
    }
  });
});
```

If no other init options remain, the file can be simplified to:

```
$(document).ready(function() {
  var app = new scout.RemoteApp();
  app.init();
});
```

To find all files that need migration, search for the text `bootstrap: {` in all **.js* files in your workspace. The files are called *index.js* by default and are usually located at *your.project.ui.html/src/main/resources/WebContent/res*.

This migration is recommended but optional. Listing all fonts to preload manually still works. To disable font preloading entirely, set the *fonts* bootstrap property to an empty array `[]`.

Radio Button Group

- The property `formFields` has been renamed to `fields` to be consistent with the Java implementation and with other composites like `GroupBox`.
- The function `selectButton` now selects the button even if it is disabled. Only the user must not select disabled buttons but the developer should still be able to do it. If you use this function, you may have to insert a check for the enabled state.

Layout of Fields in Radio Button Group and Sequence Box

Until now it was required to explicitly set grid positions for child fields of Radio Button Groups and Sequence Boxes. This was because the automatic grid layout was not yet implemented in the Scout JavaScript layer. This was no issue however if the fields have been used in connection with a Java model because then the Java layer takes care about the layout.

Now also pure JavaScript Scout applications have automatic layout for child fields of Radio Button Groups and Sequence Boxes. So if explicit grid positions (gridData.x, gridData.y) have been specified, it can be removed now as Scout takes care about it now (as it was in the Java layer already).

Form.open()

`open()` now calls `load()` first before calling `show()`. The reason is to prevent showing an empty form before any data is loaded. If you relied on the previous behavior, (e.g. if you accessed ui properties like $container right after opening the form) you would need to put that code in a function executed delayed using `form.open().then()`.

Status

The static function `scout.Status.warn()` was renamed to `scout.Status.warning()` to bring it in line

with the name of the corresponding severity constant `scout.Status.Severity.WARNING`.

MenuBar

In previous versions, right aligned menus were not stacked when there was not enough horizontal space. If this behavior is still required, the property `stackable` has to be set to *false*.

LookupRow: Removed Constructor

The custom *LookupRow* constructor with key and text parameter was removed. LookupRows must be created using the `scout.create()` object factory call.

Old style code like:

```
var lookupRow = new scout.LookupRow(data[0], data[1]);
```

must be replaced by:

```
var lookupRow = scout.create('LookupRow', {
  key: data[0],
  text: data[1],
  parentKey: data[2]  ①
});
```

① Optional parent key and other properties

Call: Skip Retries by Default (since 8.0.100)

`Call` objects no longer perform retries by default. Retries should only be active if the target service can handle repeated calls correctly.

The use retries, specify the desired number of maximal retries or a *retryIntervals* array in the model when creating the `Call` object:

```
var call1 = scout.create('AjaxCall', {
  ajaxOptions: { ... }
  maxRetries: 5
});
var call2 = scout.create('AjaxCall', {
  ajaxOptions: { ... }
  retryIntervals: [100, 200, 500, 500, 500];
});
```

ErrorHandler: Extended Support for Arbitrary Errors (since 8.0.100)

Because errors that happen in asynchronous calls (promise chains) are not delegated to the `window.onerror` handler, Scout's `ErrorHandler` class was extended. It can now be used to handle arbitrary erros in "catch" clauses or "promise fail" functions. As a consequnce, the signature of the *handle()* has been changed. It used to be hard-wired to the window.onerror event. Now it supports a flexible number of arguments.

A global instance of `ErrorHandler` is still being installed as the `window.onerror` handler. Additionally, this instance is now available as `scout.errorHandler`.

Example usages:

```
// Handling errors in "try" blocks
try {
  // ...
} catch (err) {
  scout.errorHandler.handle(err);
}

// Handle errors in jQuery AJAX calls
$.ajax(ajaxOptions)
  .done(function(data, textStatus, jqXHR) {
    /* handle success */
  })
  .fail(function(jqXHR, textStatus, errorThrown) {
    // Style 1:
    scout.errorHandler.handle(jqXHR, textStatus, errorThrown);
    // Style 2 (recommended!):
    scout.errorHandler.handle(arguments);
  })

// Handling errors in promises
doSomethingAsynchronous()
  .then(function() { /* handle result */ })
  .then(function() { /* more handling */ })
  .fail(function() {
    scout.errorHandler.handle(arguments);
  });
```

Migration:

Projects that implemented their own `ErrorHandler` must update their code.

- Instead of overriding *handle()*, override *_onWindowError()*.
- Please note that that the method *createLogMessage()* no longer exists. If you still need it, copy it from the old implementation or use *analyzeError()* to calculate a similar result.

SmartField / DateField: Renamed touch to touchMode (since 8.0.100)

Smart fields and date fields use the property `touch` to indicate that the field is running in touch mode. Unfortunately this property overrides the function `touch()` of the `FormField`. This means `touch()` cannot be used for these fields, which was not intended.

Migration: The property `touch` is actually set automatically, so most people don't have to migrate anything. If you set the property manually (e.g. to always run such a field in touch mode), you need to rename it to `touchMode`.

Other Changes

jQuery Update

The jQuery version bundled with Scout has been updated to version 3.3.1. If a *SpecRunnerMaven.html* is used to run Jasmine tests, the corresponding script tag must be updated to include `jquery-3.3.1.js` instead of `jquery-3.2.1.js`.

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