Eclipse Scout
Migration Guide

Scout Team

Version 6.0

Table of Contents

Service Release Migration . . . 1
Project Structure . . . 1
Manifest.MF . . . 2
AccessControlService . . . 2
IShellService . . . 2
Desktop . . . 3
Offline . . . 3
Mobile . . . 3
ToolButton . . . 4
Menu . . . 4
Message Box . . . 4
Table . . . 5
- Table API Changes . . . 5
- Custom Table Sorting . . . 6
- Table Field & Page . . . 7
- Changed behavior for tables with autoResizeColumns = true (since 6.0.100) . . . 8

Outline . . . 8
- Default Page selection of Outlines . . . 9

Wizard . . . 9
Form . . . 10
Form Fields . . . 10
- Validate on any Key . . . 11
- String Field . . . 12
- Button . . . 12
- Browser Field . . . 12
- Date Field . . . 13
- HTML Field . . . 13
- Tree, TreeField & TreeBox . . . 14
- Calendar, CalendarField, Planner . . . 14

Utilities . . . 14
- Cryptography . . . 15

Various API Changes . . . 16
Logging API . . . 16
Logging configuration . . . 16
Text cleanup . . . 18
Migrate to the new Job API . . . 19
- In a nutshell . . . 19
- Static accessors . . . 19

Raw Eclipse Job . . . 20
ServerJob . . . 20
ServerJob.runNow(...) . . . 21
ServerJob with other Subject . . . 21
ClientSyncJob . . . 22
ClientAsyncJob . . . 22
Delayed execution . . . 23
Repeatedly execution with a fixed delay . . . 24
Check for cancellation . . . 24
Join job . . . 25
Join job with a maximal wait time . . . 26
Join job and get the job's computation result . . . 26
Session Cookie Configuration . . . 27
Client Notifications . . . 27
Changes in a nutshell . . . 27
Publishing Notifications . . . 27
Handling Notifications . . . 28
JAX-WS Pooled Port Provider (since 6.0.300) . . . 28
Migration . . . 29
Class Renames or Moves . . . 29

This document describes what needs to be done when migrating from Eclipse Scout 5.0 to Eclipse Scout 6.0. If you are updating to a 6.0 service release, see the chapter Service Release Migration.

Service Release Migration

Neon.3 (6.0.300)

- JAX-WS Pooled Port Provider (since 6.0.300)

Neon.2 (6.0.200)

None.

Neon.1 (6.0.100)

- Changed behavior for tables with autoResizeColumns = true (since 6.0.100)

Project Structure

With the upgrade to pure maven without OSGi the project structure should be changed to the maven default [1: https://maven.apache.org/guides/getting-started/]:

Listing 1. Eclipse Plugin Structure (Scout 5.0, old)

```
org.eclipsescout.helloworld.client
    pom.xml
    plugin.xml
    src
        org
            eclipsescout
                helloworld
                    ClientSession.java
                    Activator.java

org.eclipsescout.helloworld.test
    pom.xml
    plugin.xml
    src
        org
            eclipsescout
                helloworld
                    HelloworldTest.java
```

Listing 2. Maven Project Structure (Scout 6.0, new)

```
org.eclipsescout.helloworld.client
    pom.xml
    src
        main
            java
                org
                    eclipsescout
                        helloworld
                            ClientSession.java
        test
            java
                org
                    eclipsescout
                        helloworld
                            HelloworldTest.java
```

Manifest.MF

Manifest.MF is no longer used. Migrate dependencies to `pom.xml`!

AccessControlService

The IAccessControlService has been improved to allow for other keys than the userId. AbstractAccessControlService is now generic with key as a type parameter.

If you want to use access control based on the userid as before, extend UserIdAccessControlService and change the API of execLoadPermissions to

```
protected abstract PermissionCollection execLoadPermissions(String userId)
```

IShellService

The ShellService can no longer be used because there is no access to the client side shell. Instead you can use the following code to send a document to clients:

```
BinaryResource binaryResource = new BinaryResource(templateFile.getName(), myRawData);
ClientSessionProvider.currentSession().getDesktop()
  .openUri(binaryResource, OpenUriAction.DOWNLOAD);
```

Desktop

- Renamed `IDesktop.openUrlInBrowser` to `openUri`, since passed String is not always an URL but sometimes an URI like tel:123 or mailto:foo@bar.com, etc.
- Renamed `IDesktop.openDownloadInBrowser` to `downloadResource`, added overridden methods with `BinaryResource` parameter, so it's not required to create a `IDownloadHandler` instance to use the download methods.
- Renamed `IUrlTarget` to `ITargetWindows`, `UrlTarget` to `TargetWindow`
- Renamed `DesktopEvent.TYPE_OPEN_URL_IN_BROWSER` to `TYPE_OPEN_URI`
- Renamed `DesktopEvent.TYPE_OPEN_DOWNLOAD_IN_BROWSER` to `TYPE_DOWNLOAD_RESOURCE`
- F5 Keystroke on the desktop to reload the current page is no longer necessary because the table itself now provides the f5 keystroke → Remove the keystroke from your desktop to prevent refreshing the table twice.

Offline

Offline functionality in the scout client was removed (not needed anymore). Delete - OfflineState - IClientSession.getOfflineSubject - IDesktop.changeVisibilityAfterOfflineSwitch

Mobile

org.eclipse.scout.rt.client.mobile has been merged with org.eclipse.scout.rt.client. A lot of the mobile code in that plugin has been removed because the ui is now smarter and reacts bettter to smaller screens than the previous uis.

If your project contains a mobile plugin, it is suggested to merge it with your client plugin as well.

The new mobile approach is slightly different than the one before Neon:

- There is still a device transformer which transforms the client model into a mobile optimized model. But the transformation is a lot simpler than before and is more or less limited to the adjustment of some properties.
- The transformation mainly affects mobile phones (resp. devices with a small screen size). On tablets the application will look nearly the same as on a desktop device. There are just a few optimizations made regarding touch input (e.g. the smartfield looks different).
- Previously a form based approach had been used. Every outline page was wrapped in a page form, the navigation happened by showing or hiding the corerct form. Now, the outline tree has been enhanced with mobile specific functionality. This means the navigation happens completely in the outline tree, no forms are used. There is also no need for a home form anymore which displays the available outlines. The outlines may be switched in the same way as with the desktop style. The advantage is that the mobile style (or rather the style for small devices) looks and behaves similar to the regular desktop style which should make it easier for the user.

ToolButton

- `IToolButton` has been removed, it is not necessary anymore because the desktop may now display any kind of menus.
- `IFormToolButton` has been renamed to `IFormMenu`. These menus, which display a form when selected, may now be used on any menu capable component and not only on the desktop. Therefore it has been moved from the package `org.eclipse.scout.rt.client.ui.desktop.outline` to `org.eclipse.scout.rt.client.ui.form`.

Menu

All owners of an `IContextMenu` now share a common interface: `IContextMenuOwner`. This interface provides a method `getMenuByClass(T)`, analogous to `getFieldByClass(T)`, `getColumnByClass(T)` etc.

`ITree` and `ITable` provided a similar method `getMenu(T)`. This method was deprecated in favor of `getMenuByClass(T)`.

Usually, the migration is completed by simply renaming all calls to the old method. However, it should be noted that the old behavior is not exactly reproduced in a special case: When more than one implementation of the given class T was found, the old method just returned the first instance found. The new method throws an exception in this case, because the order of the instances is not really defined. If you really want to find *any* instance of the given class, retrieve the list of all instances using getMenus() and apply the filtering by yourself.

The constructors of OutlineMenuWrapper changed. For details consult the javadoc. This was needed to ensure the correct menuTypes throughout the wrapped menu's sub-hierarchy.

The `CopyColumnsWidthsMenu` has been deleted and was replaced with a new button in `OrganizeColumnsForm`.

Message Box

- Removed title. No replacement, title is not supported anymore.
- renamed intro text to header & info/actionText to body.
- using method chaining to construct message box
  - getHiddenText() → getHiddenText()
  - setHiddenText(hiddenText) → withHiddenText(hiddenText) and returning instance of IMessageBox
- Renamed startMessageBox to show
- Removed MessageBox(String title, String introText, String okButtonText) → MessageBoxes.create().withHeader(introText).withYesButtonText(okButtonText)
- Removed MessageBox(String title, String introText, String actionText, String yesButtonText, String noButtonText, String cancelButtonText) → MessageBoxes.create().withHeader(introText).withBody(actionText)

.withYesButtonText(yesButtonText).withNoButtonText(noButtonText).cancelButtonText(cancelButtonText);

- Removed MessageBox(String title, String introText, String actionText, String yesButtonText, String noButtonText, String cancelButtonText, String hiddenText, String iconId) → MessageBoxes.create().withHeader(introText) .withBody(actionText).withYesButtonText(yesButtonText).withNoButtonText(noButtonText).withCancelButtonText(cancelButtonText).withHiddenText(hiddenText).withIconId(iconId);
- Moved MessageBox.showDeleteConfirmationMessage methods to MessagesBoxes class
- If html needs to be displayed, use the new html(IHtmlContent) method. Header / body methods do not support html.

Table

Table API Changes

- Renamed `ITable.resetDisplayableColumns()` to `resetColumns()`
- Removed `ITable.resetColumns(boolean, boolean, boolean, boolean)` from interface (is now protected in `AbstractTable`)
- `AbstractTable.execResetTable(…)`: changed signature
  - old: `protected void execResetColumns(boolean visibility, boolean order, boolean sorting, boolean widths)`
  - new: `protected void execResetColumns(Set<String> options)`
- Changed signature of `ClientUIPreferences.getTableCustomizerData`
  - From `ClientUIPreferences.getTableCustomizerData(String customizerKey)` to `ClientUIPreferences.getTableCustomizerData(ITableCustomizer customizer, String configName)`
  - From `ClientUIPreferences.setTableCustomizerData(String customizerKey, Object customizerData)` to `ClientUIPreferences.setTableCustomizerData(ITableCustomizer customizer, String configName)`
- Replaced `ITableColumnFilterManager` by `TableUserFilterManager`

  Reason for the rename is because more filter types were added. There are currently 2 filter types: Column filter and text filter, there will be a chart filter in the future. Additionally, the filtering now happens in the UI. The ui sends the filtered rows to the ui server to update its table state so that getFilteredRows return the currently visible rows on the ui. This rowsFiltered event leads to a creation of `UserTableRowFilter` which contains the filtered rows. This is the only active filter on a table. The filters managed by `TableUserFilterManager` are actually only filter states and are not added to the table.

- `AbstractColumn.execPrepareEdit(ITableRow)` must not return `null` anymore use `Cell.setEditable(boolean)` instead.
- Added `ITable#rowIconVisible` to control whether the row icon is visible or not. If set to `true` the gui creates a column which contains the row icons. The column has a fixed width, is not

moveable and always the first column (resp. the second if the table is checkable). The column is not available in the model.

If you need other settings or if you need the icon at another column position, you cannot use the row icons. Instead you have to create a column and use `Cell#setIconId(String)` to set the icons on it's cells.

If you used `ITableRow#setIconId` or `AbstractTable#getConfiguredDefaultIconId` and still want the icons to be visible, you have to set `getConfiguredRowIconVisible` to `true`.

- Refactored editable behaviour of cells.
  - `Table.isCellEditable` only returns `cell.editable` and does not consider table or row enabled and visible states. Conforms to the behaviour of the other cell properties (text, cssStyle, etc).
  - `execIsEditable` has been removed. Use `cell.setEditable` (e.g. in `execDecorateCell`) if you want a cell to behave differently than the column.
  - `decorateCellInternal` does not write properties to the cell anymore, this is now done initially or if the column property changes. Advantage: It's now possible to modify the cell properties outside of `execDecorateCell`. Furthermore, there is no need to execute this code so many times.
  - Removed `ICell.setEnabled`. Did not have any effect, use `row.setEnabled` instead. Or `ICell.setEditable` if you would like to control editability of a cell.
- `InternalTableRow` / `AbstractTable`: checked state of a row is moved to the table. The `TYPE_ROWS_UPDATED` is no longer used to notify about rows checked. Instead there is an event `TYPE_ROWS_CHECKED` which is fired when rows are checked or unchecked. Also there is a new Method on the model which is executed when rows are checked (`execRowsChecked`). This method is also available in extensions.

  A row should be set to checked from the model even if the row is disabled. For this, the method `setRowsChecked` is extended with a new parameter to identify if only enabled rows should be checked or not. The ui should only check enabled rows, so the ui-facade calls the method with true.

- Removed `ITable#rowHeightHint` / `getConfiguredRowHeightHint`

  This property was added because with rap and swt it was not possible to have variable table rows as height as their content. The only possibility to get multiline rows was to set a fixed height. This limitation is now gone. If you still want every row to have a fixed height on multiline tables, you can use css to achieve it.

Custom Table Sorting

Added `IColumn.uiSortPossible`

Sorting of table data is done by the ui whenever possible. This has the advantage, that it is faster, that less data is transferred and that it works in offline mode. The drawback is that it is not possible in every case.

Example: If an invisible column has `alwaysSortAtBegin` set to `true`, the sorting is delegated to the model. Furthermore smart columns can not be sorted by the ui because the value is not known.

If you implemented custom sorting (e.g. by overriding `AbstractColumn.compareTableRows`), you have to set `getConfiguredUiSortPossible` to `false`.

Table Field & Page

- Removed "populate status" and "selection status" methods from `IPage` and `ITableField`. The only status is on the table itself and is called "table status". `IPage` and `ITableField` have new convenience methods for getting/setting the table status (without requiring null checks on getTable()).

  Migration:

  - Replace `IPage.setPagePopulateStatus()` by `IPage.setTableStatus()`.
  - Replace `IPage.getPagePopulateStatus()` by `IPage.getTableStatus()`.
  - Properties `PROP_TABLE_SELECTION_STATUS`, `PROP_TABLE_POPULATE_STATUS`, `PROP_TABLE_STATUS_VISIBLE` no longer exist on `ITableField`. If you need to listen to them, change your listener target the the field's `ITable`.
  - Method `ITableField.createDefaultTableStatus()` was dropped without replacement. "Selection status" is not supported by Html UI at the moment (selection is visualized on the UI only, not in the model).
  - `ITableField.get/setTableStatus()` convenience methods with Strings were dropped without replacement. Use `ITableField.getTableStatus().get/setMessage()` instead.
  - `ITableField.get/setTableSelectionStatus()` were dropped without replacement. "Selection status" is not supported by Html UI at the moment (selection is visualized on the UI only, not in the model).
  - Change `ITableField.get/setTablePopulateStatus()` to `ITableField.get/setTableStatus()`
  - `ITableField.updateTableStatus()` was dropped without replacement. Simply set the table status with `ITableField.setTableStatus()`.
  - `getConfiguredTableStatusVisible()` was dropped without replacement. Instead, the initial "table status visible" property should be set on the table. (In most cases, you can simply move the `getConfiguredTableStatusVisible()` method from the table field to the table.
- Removed `AbstractPageWithTable.getConfiguredShowTableRowMenus`. Replacement: none (no functionality was provided).
- Removed `AbstractPageWithTable.getConfiguredShowEmptySpaceMenus` Replacement: if return value was false, override `computeTableEmptySpaceMenus` and return an empty list instead.
- API of `IPageWithTable` and `IPageWithNodes` merged and moved duplicate methods to `IPage`
  - `IPage` now has a `T getTable()` method, also changed abstract classes implementing these interfaces. `IPage` now expects a type parameter for the table.
  - API `IPage`:
    - added `T getTable()`

        - added `boolean isDetailFormVisible()`
        - added `void setDetailFormVisible(boolean visible)`
        - added `ITreeNode getTreeNodeFor(ITableRow tableRow)`
        - added `IPage getPageFor(ITableRow tableRow)`
        - added `ITableRow getTableRowFor(ITreeNode treeNode)`
        - added `List<ITableRow> getTableRowsFor(Collection<? extends ITreeNode> treeNodes)`
    - API `IPageWithNodes`:
        - `getInternalTable()` replaced by `getTable`
        - moved to `IPage`: `ITreeNode getTreeNodeFor(ITableRow tableRow)`
        - moved to `IPage`: `ITableRow getTableRowFor(ITreeNode childPageNode)`
    - API `IPageWithTable`:
        - moved to `IPage`: `T getTable()`
        - moved to `IPage`: `ITreeNode getTreeNodeFor(ITableRow tableRow)`
        - moved to `IPage`: `ITableRow getTableRowFor(ITreeNode childPageNode)`
        - moved to `IPage`: `List<ITableRow> getTableRowsFor(Collection<? extends ITreeNode> childPageNodes)`
- Improved page detail form handling: The detail form is now created and started when the page gets activated and closed when the page gets disposed, similar to the search form. API added `getConfiguredDetailForm`, `execInitDetailForm`, `createDetailForm`, `startDetailForm`.

  Remove the detail form handling code from `execPageActivated` / `execPageDeactivated` / `execPageDisposed` and use either `getConfiguredDetailForm` / `execInitDetailForm` or `createDetailForm`.

Changed behavior for tables with autoResizeColumns = true (since 6.0.100)

Before 6.0.1, the column width was used as weight for the calculation of the real column width. To make sure the columns don't get too small on small screens, this width is now also used as minimum / preferred width. It is not a hard minimum, the user can still make the column smaller.

So if you have tables with autoResizeColumns set to true, check the widths of the columns and adjust them if needed. The easiest way to do this is to make the screen smaller until a horizontal scrollbar appears. Then adjust the values if the column is too small and make sure the content is readable most of the time. But don't make the columns too big because you want to avoid horizontal scrollbar on large screens.

Outline

- Removed `IOutlineTableForm`, `IOutlineTreeForm` and all sub-classes. They're not supported by the new Html UI anymore.

Default Page selection of Outlines

For an Outline having a selected page is not mandatory anymore. An outline overview or the default detail form will be displayed if no page is selected. Therefore activating an outline does not automatically select the first page anymore.

If the previous behavior is still wanted, one can implemented `IDesktop.execOutlineChanged` and call `activateFirstPage` if active page is `null`.

Wizard

- Argument `containerForm` was removed. Use `getContainerForm()` instead.
- Method `decorateWizardContainerForm` was renamed to `execDecorateContainerForm` (same as `execCreateContainerForm`).

  Old code (MyWizard extends AbstractWizard):

```
@Override
protected IWizardContainerForm execCreateContainerForm() {
  MyWizardContainerForm containerForm = new MyWizardContainerForm(this);
  decorateWizardContainerForm(containerForm);
  // more custom modifications
  return containerForm;
}
```

New code:

```
@Override
protected IWizardContainerForm execCreateContainerForm() {
  return new MyWizardContainerForm(this);
}

@Override
protected void execDecorateContainerForm() {
  getContainerForm().setXyz(...);
  // more custom modifications
}
```

- Some properties were removed from IWizard:
  - `displayHint`, `displayViewId`, `modal` -→ no replacement. Set them on the wizard container form. If the wizard container form does not provide the correct value, the wizard may change them in `execDecorateContainerForm()`.
  - `iconId`, `tooltipText`, `wizardNo` -→ no replacement (legacy properties, never used).
  - `titleHtml`: use `subTitle` instead.

- `getWizard[…]Button()` methods in `IWizardContainerForm` no longer return `IButton`, but `IWizardAction`. This change allows returning menus instead of buttons. `IWizardAction` serves as a common interface for `IButton` and `IAction` and provides some methods that are commonly used for the wizard buttons (e.g. `setVisible`, `setEnabled`). Because `IAction` calls its label "text", those menus have to override getLabel/setLabel and delegate the calls to the corresponding "text" methods. Alternatively, the class `AbstractWizardMenu` may be used instead of `AbstractMenu`.
  - For own implementations of `IWizardContainerForm`, replace the return value `IButton` by `IWizardAction`.
  - For code that previously used the `setView(boolean, boolean, boolean)` method on wizard buttons, a new `setView(boolean, boolean)` method was introduced on `IButton` and `IAction` (because it does not make sense to make a button "mandatory"). This can be migrated by just deleting the third argument.

Form

- `get/setBasicTitle` removed
- `get/setSubTitle` added
- `PROP_SUB_TITLE` added
- `composeTitle` removed.
- Added default behaviour to `AbstractForm.execCreateFormData` The method now creates a new instance of the form data based on the form data annotation. Also added createFormData to the IForm interface. If `execCreateFormData` was implemented and just used the default constructor of the corresponding form data class, the method may be removed.
- Removed display-hint `IForm.DISPLAY_HINT_POPUP_DIALOG`. Not supported anymore. Use dialog or popup-window instead.

Form Fields

- Deleted `AbstractCheckBox`, `AbstractCheckboxExtension`, `ICheckBoxExtension`, `ICheckBox`.

  Use `AbstractBooleanField`, `AbstractBooleanExtension`, `IBooleanExtension`, `IBooleanField` instead.
- Renamed package `imagebox` to `imagefield` due to consistency reason.
- Deprecated `getConfiguredAutoDisplayText` in `AbstractValueField`. The display text is always updated automatically.
- Removed `AbstractDoubleField` and `AbstractDoubleColumn`. Use `AbstractBigDecimalField` and `AbstractBigDecimalColumn` instead. See Bug 464770.
- Renamed package `org.eclipse.scout.rt.client.ui.form.fields.colorpicker` to `.colorfield`.
- Removed `ContributedKeyStroke` method from all FormField classes because these are only the menus which are added on the field. Use `getMenus()` instead.
- All `AbstractExtensible*` Scout elements have been deleted. Use the normal element instead (e.g. use `AbstractStringField` instead of `AbstractExtensibleStringField`). For extension support use

the corresponding extension object (e.g. `AbstractStringFieldExtension`).

- Deleted `IMailField`, `AbstractMailField` and all associated classes and files.

  An application that requires a facility to compose an e-mail should create a form with the fields required for that application (Multiline-StringField for plain-text E-Mails, RichTextField for HTML e-mails, FileChooserField for file uploads, etc.)

- Deleted ICustomField/AbstractCustomField and all associated classes and files.
- Deleted IDocumentField/AbstractDocumentField and all associated classes and files.
- `getConfiguredTreat0AsNull` in Smartfield has been deleted. (see also Bugzilla 469902).
- Changed return value of `IGroupBox.getConfiguredScrollable` to `TriState`. Mainbox is now scrollable by default.

  Migration:

  - You can remove getConfiguredScrollable() from your mainboxes
  - If you want another groupbox to be scrollable, you have to set the groupbox to scrollable while setting the mainbox to scrollable = `false`.

- Removed `IToolButton` from forms. Therefore `IToolButtons` can not be added anymore as an extension to forms. Instead `IToolButton` can be defined inside the MainBox of a form. (`IToolButton` now is an `IMenu` and adding menus to GroupBoxes as extension is also supported.)
- Simplified form tool buttons: Refactored API to be consistent with detail and search form handling of a page. Remove the form handling code from `execStartForm` and use either `getConfiguredForm` / `execInitForm` or `createForm`.
- The search table control now gets selected if the search is required. If you had a `SearchFormToolButton`, remove the code in `Desktop.execPageSearchFormChanged`.
- When setting an inner form into an `WrappedFormField` using `setInnerForm(IForm)` the given form life cycle is handled by the wrapped form field. This means it is automatically started, disposed etc.

Validate on any Key

Replace ValidateOnAnyKey mechanism (`getConfiguredValidateOnAnyKey`) (Bug 459893):

- removed:
  - `IBasicField.setValidateOnAnyKey(boolean)`
  - `IBasicField.isValidateOnAnyKey()`
  - `IBasicField.PROP_VALIDATE_ON_ANY_KEY`
- use new updateDisplayTextOnModify-mechanism instead:
  - `IBasicField.setUpdateDisplayTextOnModify(boolean)`
  - `IBasicField.isUpdateDisplayTextOnModify(boolean)`
  - `AbstractBasicField.execChangedDisplayText()`
  - `IBasicField.PROP_UPDATE_DISPLAY_TEXT_ON_MODIFY`

- `IBasicFieldUIFacade` renamed and changed method:
  - from: `boolean setTextFromUI(String newText, boolean whileTyping)`
  - to: `void setValueFromUI(String value)`
- removed `IColorFieldUiFacade`

String Field

- Deleted `AbstractTextField`, `AbstractTextFieldExtension`, `ITextFieldExtension`, ITextField`.

  Use `AbstractStringField`, `AbstractStringExtension`, `IStringExtension`, `IStringField` instead.
- Method renaming: `getConfiguredDecorationLink()` → `getConfiguredHasAction()`.
- Method renaming: `isDecorationLink()` → `isHasAction()`.
- Method renaming: `setDecorationLink(boolean)` → `setHasAction(boolean)`.
- Method renaming/signature change: `execLinkAction(java.net.URL)` → `execAction()`. `execAction()` can access value using `getValue()`, it could create the old URL using `org.eclipse.scout.commons.IOUtility.urlTextToUrl(getValue())`.
- Removed `IStringField.isSpellCheckAsYouTypeEnabled()`, `IStringField.setSpellCheckAsYouTypeEnabled(boolean)` added `IStringField.setSpellCheckEnabled(boolean)` new ui delegates spell checker to browser, new property can be used to enable/disable spell checker for certain fields (by default it is enabled for multi-line fields, see `AbstractStringField.computeSpellCheckEnabled()`).
- Removed `IStringField.isSelectAllOnFocus()`, `IStringField.setSelectAllOnFocus(boolean)`, `IStringColumn.isSelectAllOnEdit()`, `IStringColumn.setSelectAllOnEdit(boolean)`.

Button

- The default of `getConfiguredGridUseUiWidth` was changed from true to false. This was done so that buttons are aligned with other fields by default. This only affects the grid cell, the button itself is still as width as it used to be because of `fillHorizontal = false`.

Browser Field

- `IBrowserField` is no longer a value field. The `RemoteFile` value was changed to a property of type `BinaryResource`.
  - Instead of `setValue()/getValue()` use `setBinaryResource()/getBinaryResource()`.
  - Instead of `execChangedValue()` use a `BrowserFieldListener`.
  - If you relied on the browser field to be "save needed" when setting the value (RemoteFile), you have to call `touch()` manually, because the browser field will never report "save needed" by itself (because it has no value).
- removed `AbstractBrowserField.execAcceptLocationChange`, `AbstractBrowserField.execLocationChanged`, `AbstractBrowserField.doLocationChange`. Use `AbstractBrowserField.execPostMessage` as replacement.

- Refactored `execHyperlinkAction`. With the new html ui real hyperlinks are handled by the browser. Other links (formerly local links) are now called app links. The new method `execAppLinkAction` is only called for app links, hence the parameters url and local are not necessary anymore.
  - Removed parameter url and local and renamed path to ref.
  - Renamed to `execAppLinkAction`

Date Field

- Removed the members `m_autoDate` and `m_autoTimeMillis` from `AbstractDateField`. They were replaced by a single property `PROP_AUTO_DATE` of type `java.util.Date`.

  Replace `getConfiguredAutoTimeMillis()/setAutoTimeMillis()/getAutoTimeMillis()` by `getConfiguredAutoDate()/setAutoDate()/getAutoDate()`. If both a date and time part should be set, combine them in the same `java.util.Date` argument. The methods `DateUtility.createDateTime()` and `DateUtility.convertDoubleTimeToDate()` may be useful.

- The UI facade of `AbstractDateField` was changed. To support offline, more responsive date/time validation on the new Html UI, formatting and parsing has to be performed on the UI layer, not on the model layer (otherwise, the UI would have to wait for the model on every key press).

  The date field UI facade was changed in the following way: Instead of sending a text to the model and validating/parsing it there, a already valid (from the parsing perspective) date is sent to the model. The model may then still validate it (e.g. check ranges), but the parsing is done entirely on the UI. As a consequence, not all date format patterns defined in SimpleDateFormat are supported anymore, only the most commonly used. By default, the date field uses locale-dependent patterns that are supported by the UI, see `getDefaultDateFormatPatternByLocale()` and `getDefaultTimeFormatPatternByLocale()`. Both the date and the time part of a date field have a separate pattern, because they are rendered in two separate fields on the UI.

  The method `AbstractDateField.execParseValue()` is no longer supported. It cannot be removed entirely, because it is defined on `AbstractValueField`, but is marked as deprecated and final to make it clear that it is never called. If any subclass had overridden this method, it should be deleted. The code cannot be migrated, because it is now performed in the UI only.

- Removed methods not used by the HTML UI: + remove unused execShiftDate, execShiftTime from AbstractDateField + remove unused adjustDate, adjustTime from IDateField + removed fireDateShiftActionFromUI, fireTimeShiftActionFromUI from IDateFieldUIFacade

HTML Field

- For `IHtmlField` attachments `RemoteFile` has been replaced by `BinaryResource`, therefore method signatures of `getAttachments()` and `setAttachments` have changed.
  - Replace `RemoteFile` with `BinaryResource`.
  - Attachments must be used within the `IHtmlField's value as `src="binaryResource:test.png"` (instead of `src="test.png"`). Append `binaryResource:` prefix where attachments are used.

    - New feature: Icons can be used without adding them as attachment using `src="iconid:ApplicationLogo"`.
    - New property for selection tracking, changeable with methods `isSelectionTrackingEnabled()` and `setSelectionTrackingEnabled(boolean)`. Selection tracking with `getSelectionStart()` and `getSelectionEnd()` is only possible when selection tracking is enabled.
- Removed html editor support on html field. If you used a html editor you can create a custom field an include an existing html editor.

Tree, TreeField & TreeBox

- If all child nodes of a node in a tree are deleted, a `TreeEvent` with the new type `ALL_CHILD_NODES_DELETED` is fired (instead of `NODES_DELETED`). This is useful for optimization.

  If you previously added a listener for the type `NODES_DELETED`, you have to check if your implementation needs to listen to the new `ALL_CHILD_NODES_DELETED` as well.

- `AbstractTreeNode` / `AbstractTree` / `AbstractTreeBox`: checked state of a row is moved to the tree. The `TYPE_NODE_UPDATED` is no longer used to notify about node checked. Instead there is an event `TYPE_NODES_CHECKED` which is fired when nodes are checked or unchecked. Also there is a new Method on the model which is executed when nodes are checked (`execNodessChecked`). This method is also available in extensions.

  Also the implementation to check child nodes of a tree when a parent is checked is moved from the `AbstractTreeBox` to the tree. But the configuration can be done on the `AbstractTreeBox`. A node should be set to checked from the model even if the node is disabled. For this, the method `setNodesChecked` is extended with a new param to identify if only enabled nodes should be checked or not. The ui should only check enabled nodes, so the ui-facade calls the method with true.

Calendar, CalendarField, Planner

- Moved display-mode constants from `ICalender` and `IPlanner` to separate interface classes and let `IPlannerDisplayMode` extend `ICalenderDisplayMode` because they share some constants.
- Removed `get/setColor()` from `ICalenderItem`, replaced with `get/setCssClass()`.
- Removed `decorateCell/-Internal` method from `AbstractCalendarItemProvider`
- Moved `get/setExternalKey()` from `ICalendarAppointment` to base class `ICalenderItem`.
- Removed cell instance from `CalendarComponent`

Utilities

- Removed methods `UserAgentUtility.isRichClient()` and `.isWebClient()`.
- `HTMLUtility` has been deprecated. There is no replacement.
- `NumberUtility.sum(double…)` -→ use `sum(Number…)`

- `NumberUtility.sum(long…)` -→ use `sum(Number…)`
- Removed `NumberUtility.avg(double…)`
- Removed `NumberUtility.divide(double, double)`
  - `NlsUtility.getDefaultLocale()` has been removed -→ use `NlsLocale.CURRENT`
- The following Classes have been moved. Organize imports to fix errors:
  - `IDNDSupport`
  - `TransferObject`
  - `TextTransferObject`
  - `ResourceListTransferObject`
  - `JavaTransferObject`
  - `ImageTransferObject`
  - All Classes that once existed in `org.eclipse.scout.commons.*`. Most of them have been moved to `org.eclipse.scout.rt.platform.*`.
- Renamed `FileListTransferObject` to `ResourceListTransferObject`
- Removed `isText()`, `isFileList()`, `isImage()`, `isLocalObject()` from `TransferObject`. Replacement: `instanceof` check for the appropriate subclasses of `TransferObject`.
- Removed `TextTransferObject(String plainText, String htmlText)` and `TextTransferObject.getHtmlText()`. See Bug 465797.
- Moved `MultiClientSessionCookieStore` to `org.eclipse.scout.rt.servicetunnel` and renamed it to `MultiSessionCookieStore`. It can now be used in client and server environments.

  To make the service tunnel work with multiple sessions over HTTP, the `MultiSessionCookieStore` has to be installed. This is not done automatically, because the cookie manager is global for the entire JVM. Overriding this global variable may break things in a JEE environment with multiple applications or a pre-installed custom cookie manager. There are two options to install Scout's `MultiSessionCookieStore`:

  - Set the default cookie manager programmatically somewhere in your code. This is the way provided by the JVM, see http://docs.oracle.com/javase/tutorial/networking/cookies/cookiemanager.html for details.
  - Use Scout's auto-install mechanism by setting the property `org.eclipse.scout.rt.servicetunnel.multiSessionCookieStoreEnabled` in your `config.properties` to `true`. This is the recommended way.

Cryptography

`EncryptionUtility`, `PublicKeyUtility`, `TripleDES` have been deprecated because these classes use insecure cryptography. Use the new `SecurityUtility` or the Java Cryptography Architecture instead [2: http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html].

When changing the cryptography algorithms in you application please keep in mind that all existing encrypted, hashed or signed data becomes invalid! Consider migrating these data first.

Various API Changes

- Changed `ILookupRow` to fluent API: use with... instead of set...
- `IClientSession.stopSession()` was renamed to `stop()` to match `IServerSession.stop()`.
- Deleted validation rule infrastructure: Deleted package org.eclipse.scout.rt.shared.validate with all subpackages and the containing classes. Furthermore the class org.eclipse.scout.rt.shared.data.form.ValidationRule has been deleted.
- `UiLayer`: Removed values `JSP`, `JSF`, `RAP`, `SWING` and added value `HTML`.
- `UserAgentUtility`: API removed `isRapUi()`, `isSwingUi()`
- The unused, obsolete classes `org.eclipse.scout.rt.client.ui.form.fields.ValueFieldEvent` and `org.eclipse.scout.rt.client.ui.form.fields.ValueFieldListener` were removed.

Logging API

Scout switched from a custom, typically java.util.logging-based logger implementation to SLF4j. The log format does not support indexed placeholders anymore.

The regular expression pattern `\{\d+\}` finds potential occurrences. Replace those within log formats with `{}`. See SLF4j MessageFormatter.

Listing 3. Placeholders in log format

```
LOG.info("message {}", obj); // this worked before and still works. No action required

LOG.info("message {0}", obj); // the index is not supported anymore. You have to
remove it (see previous statement)
```

Indexed placeholders are actually deprecated since Scout's open-source debut. The values were filled in from left to right, independent of the possibly declared index.

Logging configuration

migrate logging.properties to logback.xml

1) in logging.properties apply the following regex replacements:

```
search: ^(\w.*)\.level\s*=\s*(ALL|OFF|SEVERE|WARNING|INFO|FINE|FINEST)\s*$
replace: <logger name="$1" level="$2"/>

search: ^(\w.*)\.useParentHandlers\s*=\s*(false)\s*$
replace: <logger name="$1"><appender-ref ref="CONSOLE"/></logger>

search: ^#+\s*(.*)$
replace: <!-- $1 -->

search: (FINEST|finest)
replace: TRACE

search: (FINE|fine)
replace: DEBUG

search: (WARNING|warning)
replace: WARN

search: (SEVERE|severe)
replace: ERROR
```

2) create a new logback.xml as

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>

  <appender name="CONSOLE"
    class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} %-5level %logger{36} - %msg %n</pattern>
    </encoder>
  </appender>

  <root level="INFO">
    <appender-ref ref="CONSOLE" />
  </root>
  <!-- (3) -->

</configuration>
```

3) include the converted content of logging.properties at 1.

4) adjust the format pattern if needed

available variables are

```
%d{HH:mm:ss.SSS}
%thread
%-5level
%logger{36}
%msg
%n
%X{scout.ui.session.id}
%X{scout.session.id}
%X{http.request.method}
%X{http.request.uri}
%X{http.session.id}
%X{scout.user.name}
%X{subject.principal.name}
```

Default ui.html pattern

```
<pattern>%d{HH:mm:ss.SSS} %-5level %logger{36} - %msg [%X{subject.principal.name} @
%X{http.request.method} %X{http.request.uri} %X{scout.ui.session.id}]%n</pattern>
```

Default server pattern

```
<pattern>%d{HH:mm:ss.SSS} %-5level %logger{36} - %msg [%X{subject.principal.name} @
%X{http.session.id} in %thread ]%n</pattern>
```

Text cleanup

All unused texts in ScoutTextProviderService were removed. If you were using one of the deleted ones, you find them in:

- ScoutTexts_bg.properties
- ScoutTexts_cs.properties
- ScoutTexts_da.properties
- ScoutTexts_de_DE.properties
- ScoutTexts_de.properties
- ScoutTexts_el.properties
- ScoutTexts_es.properties
- ScoutTexts_fi.properties
- ScoutTexts_fr_BE.properties
- ScoutTexts_fr.properties
- ScoutTexts_hr.properties
- ScoutTexts_hu.properties

- ScoutTexts_it.properties
- ScoutTexts_ja.properties
- ScoutTexts_nl_BE.properties
- ScoutTexts_nl.properties
- ScoutTexts_no.properties
- ScoutTexts_pl.properties
- ScoutTexts_pt_br.properties
- ScoutTexts_ru.properties
- ScoutTexts_se.properties
- ScoutTexts_sk.properties
- ScoutTexts_sl.properties
- ScoutTexts_sr.properties
- ScoutTexts_tr.properties
- ScoutTexts_zh_TW.properties
- ScoutTexts_zh.properties
- ScoutTexts.properties

Migrate to the new Job API

Eclipse jobs are replaced by Scout Job Manager API.

In a nutshell

Scout provides a job manager based on Java Executors framework to run tasks in parallel, and on Quartz Trigger API to support for schedule plans. A task (aka job) can be scheduled to commence execution either immediately upon being scheduled, or delayed some time in the future. A job can be single executing, or recurring based on some schedule plan.

A job is defined as some work to be executed asynchronously and is associated with a `JobInput` to describe how to run that work. The work is given to the job manager in the form of a `Runnable` or `Callable`. The only difference is, that a `Runnable` represents a 'fire-and-forget' action, meaning that the submitter of the job does not expect the job to return a result. On the other hand, a `Callable` returns the computation's result, which the submitter can await for. Of course, a runnable's completion can also be waited for.

See Scout architecture documentation for more information.

Static accessors

- ServerJob.getCurrentSession() → ServerSessionProvider.currentSession()
- ClientJob.getCurrentSession() → ClientSessionProvider.currentSession()

- ServerJob.isCurrentJobCancelled() → RunMonitor.CURRENT.get().isCancelled()

Raw Eclipse Job

Listing 4. before Scout 'N' release (<=5.0.x)

```
new Job("job-name") {

  @Override
  protected IStatus run(IProgressMonitor monitor) {
    // do something
  }
}.schedule();
```

Listing 5. since Scout 'N' release (>=5.1.x)

```
Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, Jobs.newInput()
    .withName("job-name"));
```

ServerJob

Listing 6. before Scout 'N' release (<=5.0.x)

```
new ServerJob("job-name", ServerJob.getCurrentSession()) {

@Override
protected IStatus runTransaction(IProgressMonitor monitor) throws Exception {
  // do something
  return Status.OK_STATUS;
}
}.schedule();
```

Listing 7. since Scout 'N' release (>=5.1.x)

```
Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, Jobs.newInput()
    .withRunContext(ServerRunContexts.copyCurrent())
    .withName("job-name"));
```

ServerJob.runNow(...)

Listing 8. before Scout 'N' release (<=5.0.x)

```
new ServerJob("job-name", ServerJob.getCurrentSession()) {

@Override
protected IStatus runTransaction(IProgressMonitor monitor) throws Exception {
  // do something
  return Status.OK_STATUS;
}
}.runNow(new NullProgressMonitor());
```

Listing 9. since Scout 'N' release (>=5.1.x)

```
ServerRunContexts.copyCurrent().run(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
});
```

ServerJob with other Subject

Listing 10. before Scout 'N' release (<=5.0.x)

```
new ServerJob("job-name", ServerJob.getCurrentSession(), subject) {

@Override
protected IStatus runTransaction(IProgressMonitor monitor) throws Exception {
  // do something
  return Status.OK_STATUS;
}
}.schedule();
```

Listing 11. since Scout 'N' release (>=5.1.x)

```
Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, Jobs.newInput()
    .withName("job-name")
    .withRunContext(ServerRunContexts.copyCurrent()
        .withSubject(subject)));
```

ClientSyncJob

Listing 12. before Scout 'N' release (<=5.0.x)

```
new ClientSyncJob("job-name", ClientSessionProvider.currentSession()) {

  @Override
  protected void runVoid(IProgressMonitor monitor) throws Throwable {
    // do something
  }
}.schedule();
```

Listing 13. since Scout 'N' release (>=5.1.x)

```
ModelJobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, ModelJobs
    .newInput(ClientRunContexts.copyCurrent())
    .withName("job-name"));
```

ClientAsyncJob

Listing 14. before Scout 'N' release (<=5.0.x)

```
new ClientAsyncJob("job-name", ClientSessionProvider.currentSession()) {

  @Override
  protected void runVoid(IProgressMonitor monitor) throws Throwable {
    // do something
  }
}.schedule();
```

Listing 15. since Scout 'N' release (>=5.1.x)

```
Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, Jobs.newInput()
    .withRunContext(ClientRunContexts.copyCurrent())
    .withName("job-name"));
```

Delayed execution

Listing 16. before Scout 'N' release (<=5.0.x)

```
new Job("job-name") {

@Override
protected IStatus run(IProgressMonitor monitor) {
  // do something
}
}.schedule(5_000);
```

Listing 17. since Scout 'N' release (>=5.1.x)

```
Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, Jobs.newInput()
    .withName("job-name")
    .withExecutionTrigger(Jobs.newExecutionTrigger()
        .withStartIn(5, TimeUnit.SECONDS)));
```

Repeatedly execution with a fixed delay

Listing 18. before Scout 'N' release (<=5.0.x)

```
new Job("job-name") {

@Override
protected IStatus run(IProgressMonitor monitor) {
  // do something
  schedule(5_000);
}
}.schedule(5_000);
```

Listing 19. since Scout 'N' release (>=5.1.x)

```
Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, Jobs.newInput()
    .withName("job-name")
    .withExecutionTrigger(Jobs.newExecutionTrigger()
        .withSchedule(FixedDelayScheduleBuilder.repeatForever(5, TimeUnit.SECONDS))
        .withStartIn(5, TimeUnit.SECONDS)));
```

Check for cancellation

Listing 20. before Scout 'N' release (<=5.0.x)

```
new Job("job-name") {

@Override
protected IStatus run(IProgressMonitor monitor) {
  // do first chunk of work
  if (monitor.isCanceled()) {
      return Status.CANCEL_STATUS;
  }
  // do second chunk of work
  if (monitor.isCanceled()) {
    return Status.CANCEL_STATUS;
  }
  // do third chunk of work
  return Status.OK_STATUS;
}
}.schedule();
```

Listing 21. since Scout 'N' release (>=5.1.x)

```
Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do first chunk of work
    if (RunMonitor.CURRENT.get().isCancelled()) {
      return;
    }
    // do second chunk of work
    if (RunMonitor.CURRENT.get().isCancelled()) {
      return;
    }
    // do third chunk of work
  }
}, Jobs.newInput()
    .withName("job-name"));
```

Join job

Listing 22. before Scout 'N' release (<=5.0.x)

```
Job job = new Job("job-name") {

  @Override
  protected IStatus run(IProgressMonitor monitor) {
    // do something
    return Status.OK_STATUS;
  }
};
job.schedule();
job.join();
```

Listing 23. since Scout 'N' release (>=5.1.x)

```
IFuture<Void> future = Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, Jobs.newInput()
    .withName("job-name"));

future.awaitDone();
```

Join job with a maximal wait time

Listing 24. before Scout 'N' release (<=5.0.x)

```
Job job = new Job("job-name") {

  @Override
  protected IStatus run(IProgressMonitor monitor) {
    // do something
    return Status.OK_STATUS;
  }
};
job.schedule();
job.join(5_000, new NullProgressMonitor());
```

Listing 25. since Scout 'N' release (>=5.1.x)

```
IFuture<Void> future = Jobs.schedule(new IRunnable() {

  @Override
  public void run() throws Exception {
    // do something
  }
}, Jobs.newInput()
    .withName("job-name"));

future.awaitDone(5, TimeUnit.SECONDS);
```

Join job and get the job's computation result

Listing 26. before Scout 'N' release (<=5.0.x)

```
final AtomicReference<String> result = new AtomicReference<>();

Job job = new Job("job-name") {

  @Override
  protected IStatus run(IProgressMonitor monitor) {
    // do something
    result.set("abc");
    return Status.OK_STATUS;
  }
};
job.schedule();
job.join();
System.out.println(result);
```

Listing 27. since Scout 'N' release (>=5.1.x)

```
IFuture<String> future = Jobs.schedule(new Callable<String>() {

  @Override
  public String call() throws Exception {
    // do something
    return "result";
  }
}, Jobs.newInput()
    .withName("job-name"));

String result = future.awaitDoneAndGet();
System.out.println(result);
```

Session Cookie Configuration

The Scout HTML UI Session cookie requires some security flags. Please refer to the Scout documentation chapter "Session Cookie (JSESSIONID Cookie) Configuration" to learn how to configure your JSESSIONID cookie.

Client Notifications

Check out the docs for a description about client notifications.

Changes in a nutshell

- There is only one poller per client (instead of per session): `ClientNotificationPoller`
- Long polling is used instead of polling in regular intervals
- Client notifications are plain serializable objects and do not need to implement the interface `IClientNotification` anymore
- `ClientNotificationRegistry` is used to register client notifications instead of `IClientNotificationService`
- If a notification needs to be handled temporarily, `AbstractObservableNotificationHandler` can be used to register a listener
- If a notification needs to be handled always, a handler can be created as subtypes of `INotificationHandler<T extends Serializable>` to always handle messages of type `T` `instead of creating a `IClientNotificationConsumerListener`
- The method `coalesce` on the client notification is replaced with a class of type `ICoalescer<T>`
- ServiceTunnel is now a bean instead of a member of the client session

Publishing Notifications

Listing 28. before Scout 'N' release (<=5.0.x)

```
SERVICES.getService(IClientNotificationService.class)
    .putNotification(new UserChangedClientNotification(userId), new
UserKeyClientNotificationFilter(userId, 60000L));
```

Listing 29. since Scout 'N' release (>=5.1.x)

```
String userId = "testUser";
BEANS.get(ClientNotificationRegistry.class)
    .putForUser(userId, new UserChangedClientNotification(userId));
```

Handling Notifications

Listing 30. before Scout 'N' release (<=5.0.x)

```
IClientNotificationConsumerListener m_userChangedNotificationListener = new
IClientNotificationConsumerListener() {
@Override
public void handleEvent(ClientNotificationConsumerEvent event, boolean sync) {
  if (event.getClientNotification() instanceof UserChangedClientNotification) {
    //handle ...
  }
}
};
SERVICES.getService(IClientNotificationConsumerService.class)
    .addClientNotificationConsumerListener(AbstractCoreClientSession.get(),
m_userChangedNotificationListener);
```

Listing 31. since Scout 'N' release (>=5.1.x)

```
class UserChangedClientNotificationHandler implements INotificationHandler
<UserChangedClientNotification> {

  @Override
  public void handleNotification(UserChangedClientNotification notification) {
    //handle ...
  }
}
```

JAX-WS Pooled Port Provider (since 6.0.300)

Creating web service and port instances are expensive operations (at least if the reference implementation Metro or the one bundled with the JRE is used). Especially parsing the WSDL and XSD files as well as building JAXB contexts and it is even worse if they are performed in parallel (due to synchronization).

The `PooledPortProvider` is the new default strategy for creating ports. Actually the pooled provider uses two pools, one for service instances and another for port instances (which are created by a service instance). A Scout transaction member keeps track of leased ports and puts them back into the pool when the Scout transaction releases its resources. Further, the transaction member ensures that the same port is used within a transaction, once it has been leased.

Port instances are reset when they are put back into the pool. Some JAX-WS implementations provide a suitable operation for resetting the port (i.e. Metro as well as the RI bundled with Java 8). Otherwise the request context is cleansed as good as possible. The corresponding `JaxWsImplementorSpecifics.resetRequestContext(Object)` can be extended to customize the cleansing.

The `AbstractWebServiceClient` does not distinguish between `PortProducer` and `PortCache` anymore. Both are `IPortProvider` strategies and the new `PooledPortPorvider` is just another one, frankly the new default. Setting the configuration property `jaxws.consumer.portPool.enabled` to `false` disables the pool and enables the previous behavior (bare `PortProducer`, wrapped by a `PortCache` instance that stashes ports).

The internal state of the pools is reported on the diagnostics servlet.

Migration

The following method has been renamed and the return type has been changed to `IPortProvider`. More important it returns a pooled, cached, bare provider or uses any other strategy. In other words, the `AbstractWebServiceClient` does not wrap the producer into a `PortCache` anymore.

```
class: org.eclipse.scout.rt.server.jaxws.consumer.AbstractWebServiceClient
old:   getConfiguredPortProducer(Class<SERVICE>, Class<PORT>, URL, String, String,
IPortInitializer)
new:   getConfiguredPortProvider(Class<SERVICE>, Class<PORT>, URL, String, String,
IPortInitializer)
                        ^^^^^^^^
```

Class Renames or Moves

Excluding Tests

5.1.x	5.2.x/6.0.x
org.eclipse.scout.commons.annotations.ColumnData.java	org.eclipse.scout.rt.client.dto.ColumnData.java
org.eclipse.scout.commons.annotations.ConfigProperty.java	org.eclipse.scout.rt.platform.annotations.ConfigProperty.java
org.eclipse.scout.commons.annotations.FormData.java	org.eclipse.scout.rt.client.dto.FormData.java
org.eclipse.scout.commons.annotations.InjectFieldTo.java	org.eclipse.scout.rt.platform.extension.InjectFieldTo.java

5.1.x	5.2.x/6.0.x
org.eclipse.scout.commons.annotations.Internal.java	org.eclipse.scout.rt.platform.annotations.Internal.java
org.eclipse.scout.commons.annotations.IOrdered.java	org.eclipse.scout.rt.platform.IOrdered.java
org.eclipse.scout.commons.annotations.OrderedCollection.java	org.eclipse.scout.rt.platform.util.collection.OrderedCollection.java
org.eclipse.scout.commons.annotations.OrderedComparator.java	org.eclipse.scout.rt.platform.OrderedComparator.java
org.eclipse.scout.commons.annotations.PageData.java	org.eclipse.scout.rt.client.dto.PageData.java
org.eclipse.scout.commons.ArrayComparator.java	org.eclipse.scout.rt.platform.util.ArrayComparator.java
org.eclipse.scout.commons.Base64Utility.java	org.eclipse.scout.rt.platform.util.Base64Utility.java
org.eclipse.scout.commons.beans.AbstractPropertyObserver.java	org.eclipse.scout.rt.platform.reflect.AbstractPropertyObserver.java
org.eclipse.scout.commons.beans.BasicPropertySupport.java	org.eclipse.scout.rt.platform.reflect.BasicPropertySupport.java
org.eclipse.scout.commons.beans.FastBeanInfo.java	org.eclipse.scout.rt.platform.reflect.FastBeanInfo.java
org.eclipse.scout.commons.beans.FastBeanUtility.java	org.eclipse.scout.rt.platform.reflect.FastBeanUtility.java
org.eclipse.scout.commons.beans.FastPropertyDescriptor.java	org.eclipse.scout.rt.platform.reflect.FastPropertyDescriptor.java
org.eclipse.scout.commons.BeanUtility.java	org.eclipse.scout.rt.platform.util.BeanUtility.java
org.eclipse.scout.commons.CellRange.java	org.eclipse.scout.rt.platform.util.CellRange.java
org.eclipse.scout.commons.ClassIdentifier.java	org.eclipse.scout.rt.platform.classid.ClassIdentifier.java
org.eclipse.scout.commons.CollationRulesPatch.java	org.eclipse.scout.rt.platform.nls.CollationRulesPatch.java
org.eclipse.scout.commons.CollectionUtility.java	org.eclipse.scout.rt.platform.util.CollectionUtility.java
org.eclipse.scout.commons.CollectorVisitor.java	org.eclipse.scout.rt.platform.visitor.CollectorVisitor.java
org.eclipse.scout.commons.ColorUtility.java	org.eclipse.scout.rt.platform.util.ColorUtility.java
org.eclipse.scout.commons.CompareUtility.java	org.eclipse.scout.rt.platform.util.CompareUtility.java
org.eclipse.scout.commons.CompositeObject.java	org.eclipse.scout.rt.platform.util.CompositeObject.java

5.1.x	5.2.x/6.0.x
org.eclipse.scout.commons.ConfigUtility.java	org.eclipse.scout.rt.platform.config.ConfigUtility.java
org.eclipse.scout.commons.CSSPatch.java	org.eclipse.scout.rt.platform.html.CSSPatch.java
org.eclipse.scout.commons.dnd.JavaTransferObject.java	org.eclipse.scout.rt.client.ui.dnd.JavaTransferObject.java
org.eclipse.scout.commons.EncryptionUtility.java	org.eclipse.scout.rt.platform.security.EncryptionUtility.java
org.eclipse.scout.commons.eventlistprofiler.EventListenerProfiler.java	org.eclipse.scout.rt.platform.eventlistprofiler.EventListenerProfiler.java
org.eclipse.scout.commons.eventlistprofiler.EventListenerSnapshot.java	org.eclipse.scout.rt.platform.eventlistprofiler.EventListenerSnapshot.java
org.eclipse.scout.commons.exception.InitializationException.java	org.eclipse.scout.rt.platform.exception.InitializationException.java
org.eclipse.scout.commons.exception.PlaceholderException.java	org.eclipse.scout.rt.platform.exception.PlaceholderException.java
org.eclipse.scout.commons.FileUtility.java	org.eclipse.scout.rt.platform.util.FileUtility.java
org.eclipse.scout.commons.holders.Holder.java	org.eclipse.scout.rt.platform.holders.Holder.java
org.eclipse.scout.commons.holders.IBeanArrayHolder.java	org.eclipse.scout.rt.platform.holders.IBeanArrayHolder.java
org.eclipse.scout.commons.holders.ITableBeanRowHolder.java	org.eclipse.scout.rt.platform.holders.ITableBeanRowHolder.java
org.eclipse.scout.commons.holders.NVPair.java	org.eclipse.scout.rt.platform.holders.NVPair.java
org.eclipse.scout.commons.holders.TableBeanHolderFilter.java	org.eclipse.scout.rt.platform.holders.TableBeanHolderFilter.java
org.eclipse.scout.commons.holders.TableHolderFilter.java	org.eclipse.scout.rt.platform.holders.TableHolderFilter.java
org.eclipse.scout.commons.html.HtmlBinds.java	org.eclipse.scout.rt.platform.html.HtmlBinds.java
org.eclipse.scout.commons.html.IHtmlElement.java	org.eclipse.scout.rt.platform.html.IHtmlElement.java
org.eclipse.scout.commons.html.IHtmlInput.java	org.eclipse.scout.rt.platform.html.IHtmlInput.java
org.eclipse.scout.commons.html.IHtmlListElement.java	org.eclipse.scout.rt.platform.html.IHtmlListElement.java
org.eclipse.scout.commons.html.IHtmlTable.java	org.eclipse.scout.rt.platform.html.IHtmlTable.java
org.eclipse.scout.commons.html.IHtmlTableCell.java	org.eclipse.scout.rt.platform.html.IHtmlTableCell.java

5.1.x	5.2.x/6.0.x
org.eclipse.scout.commons.html.IHtmlTableRow.java	org.eclipse.scout.rt.platform.html.IHtmlTableRow.java
org.eclipse.scout.commons.html.internal.EmptyHtmlNodeBuilder.java	org.eclipse.scout.rt.platform.html.internal.EmptyHtmlNodeBuilder.java
org.eclipse.scout.commons.html.internal.HtmlContentBuilder.java	org.eclipse.scout.rt.platform.html.internal.HtmlContentBuilder.java
org.eclipse.scout.commons.html.internal.HtmlNodeBuilder.java	org.eclipse.scout.rt.platform.html.internal.HtmlNodeBuilder.java
org.eclipse.scout.commons.html.internal.HtmlTableDataBuilder.java	org.eclipse.scout.rt.platform.html.internal.HtmlTableDataBuilder.java
org.eclipse.scout.commons.HTMLUtility.java	org.eclipse.scout.rt.platform.html.HTMLUtility.java
org.eclipse.scout.commons.index.AbstractMultiValueIndex.java	org.eclipse.scout.rt.platform.index.AbstractMultiValueIndex.java
org.eclipse.scout.commons.index.AbstractSingleValueIndex.java	org.eclipse.scout.rt.platform.index.AbstractSingleValueIndex.java
org.eclipse.scout.commons.index.IIndex.java	org.eclipse.scout.rt.platform.index.IIndex.java
org.eclipse.scout.commons.index.IndexedStore.java	org.eclipse.scout.rt.platform.index.IndexedStore.java
org.eclipse.scout.commons.internal.tripledes.TripleDES.java	org.eclipse.scout.rt.platform.security.TripleDES.java
org.eclipse.scout.commons.LocaleUtility.java	org.eclipse.scout.rt.platform.nls.LocaleUtility.java
org.eclipse.scout.commons.logger.LevelRangeFilter.java	org.eclipse.scout.rt.platform.logger.LevelRangeFilter.java
org.eclipse.scout.commons.mail.CharsetSafeMimeMessage.java	org.eclipse.scout.rt.shared.mail.CharsetSafeMimeMessage.java
org.eclipse.scout.commons.MatrixUtility.java	org.eclipse.scout.rt.platform.util.MatrixUtility.java
org.eclipse.scout.commons.nls.DynamicNls.java	org.eclipse.scout.rt.platform.nls.DynamicNls.java
org.eclipse.scout.commons.nls.NlsLocale.java	org.eclipse.scout.rt.platform.nls.NlsLocale.java
org.eclipse.scout.commons.nls.NlsResourceBundleCache.java	org.eclipse.scout.rt.platform.nls.NlsResourceBundleCache.java
org.eclipse.scout.commons.nls.NlsUtility.java	org.eclipse.scout.rt.platform.nls.NlsUtility.java
org.eclipse.scout.commons.parsers.IntoParser.java	org.eclipse.scout.rt.server.jdbc.parsers.IntoParser.java
org.eclipse.scout.commons.parsers.sql.SqlParser.java	org.eclipse.scout.rt.server.jdbc.parsers.sql.SqlParser.java
org.eclipse.scout.commons.parsers.sql.SqlParserToken.java	org.eclipse.scout.rt.server.jdbc.parsers.sql.SqlParserToken.java

5.1.x	5.2.x/6.0.x
org.eclipse.scout.commons.parsers.token.DatabaseSpecificToken.java	org.eclipse.scout.rt.server.jdbc.parsers.token.DatabaseSpecificToken.java
org.eclipse.scout.commons.parsers.token.FunctionInputToken.java	org.eclipse.scout.rt.server.jdbc.parsers.token.FunctionInputToken.java
org.eclipse.scout.commons.parsers.token.TextToken.java	org.eclipse.scout.rt.server.jdbc.parsers.token.TextToken.java
org.eclipse.scout.commons.parsers.token.ValueInputToken.java	org.eclipse.scout.rt.server.jdbc.parsers.token.ValueInputToken.java
org.eclipse.scout.commons.parsers.token.ValueOutputToken.java	org.eclipse.scout.rt.server.jdbc.parsers.token.ValueOutputToken.java
org.eclipse.scout.commons.PropertiesHelper.java	org.eclipse.scout.rt.platform.config.PropertiesHelper.java
org.eclipse.scout.commons.PublicKeyUtility.java	org.eclipse.scout.rt.platform.security.PublicKeyUtility.java
org.eclipse.scout.commons.Range.java	org.eclipse.scout.rt.platform.util.Range.java
org.eclipse.scout.commons.ReflectionUtility.java	org.eclipse.scout.rt.platform.reflect.ReflectionUtility.java
org.eclipse.scout.commons.resource.MimeType.java	org.eclipse.scout.rt.platform.resource.MimeType.java
org.eclipse.scout.commons.RFCWrapperPart.java	org.eclipse.scout.rt.shared.mail.RFCWrapperPart.java
org.eclipse.scout.commons.security.SimplePrincipal.java	org.eclipse.scout.rt.platform.security.SimplePrincipal.java
org.eclipse.scout.commons.SecurityUtility.java	org.eclipse.scout.rt.platform.security.SecurityUtility.java
org.eclipse.scout.commons.serialization.BasicObjectSerializer.java	org.eclipse.scout.rt.platform.serialization.BasicObjectSerializer.java
org.eclipse.scout.commons.serialization.IObjectReplacer.java	org.eclipse.scout.rt.platform.serialization.IObjectReplacer.java
org.eclipse.scout.commons.serialization.IObjectSerializer.java	org.eclipse.scout.rt.platform.serialization.IObjectSerializer.java
org.eclipse.scout.commons.serialization.IObjectSerializerFactory.java	org.eclipse.scout.rt.platform.serialization.IObjectSerializerFactory.java
org.eclipse.scout.commons.status.IMultiStatus.java	org.eclipse.scout.rt.platform.status.IMultiStatus.java
org.eclipse.scout.commons.status.IStatus.java	org.eclipse.scout.rt.platform.status.IStatus.java
org.eclipse.scout.commons.status.MultiStatus.java	org.eclipse.scout.rt.platform.status.MultiStatus.java
org.eclipse.scout.commons.status.Status.java	org.eclipse.scout.rt.platform.status.Status.java
org.eclipse.scout.commons.StringUtility.java	org.eclipse.scout.rt.platform.util.StringUtility.java

5.1.x	5.2.x/6.0.x
org.eclipse.scout.commons.ToStringBuilder.java	org.eclipse.scout.rt.platform.util.ToStringBuilder.java
org.eclipse.scout.commons.TriState.java	org.eclipse.scout.rt.platform.util.TriState.java
org.eclipse.scout.commons.TuningUtility.java	org.eclipse.scout.rt.platform.util.TuningUtility.java
org.eclipse.scout.commons.TypeCastUtility.java	org.eclipse.scout.rt.platform.util.TypeCastUtility.java
org.eclipse.scout.commons.VerboseUtility.java	org.eclipse.scout.rt.platform.util.VerboseUtility.java
org.eclipse.scout.commons.XmlUtility.java	org.eclipse.scout.rt.platform.util.XmlUtility.java
org.eclipse.scout.rt.client.ui.form.fields.imagebox.IImageField.java	org.eclipse.scout.rt.client.ui.form.fields.imagefield.IImageField.java
org.eclipse.scout.rt.client.ui.form.fields.imagebox.ImageFieldEvent.java	org.eclipse.scout.rt.client.ui.form.fields.imagefield.ImageFieldEvent.java
org.eclipse.scout.rt.platform.service.internal.AbstractHolderArgumentVisitor.java	org.eclipse.scout.rt.shared.servicetunnel.internal.AbstractHolderArgumentVisitor.java
org.eclipse.scout.rt.platform.util.csv.ArrayConsumer.java	org.eclipse.scout.rt.shared.csv.ArrayConsumer.java
org.eclipse.scout.rt.platform.util.DateFormatProvider.java	org.eclipse.scout.rt.platform.util.date.DateFormatProvider.java
org.eclipse.scout.rt.platform.util.DateUtility.java	org.eclipse.scout.rt.platform.util.date.DateUtility.java
org.eclipse.scout.rt.server.commons.servlet.filter.authentication.PathInfoFilter.java	org.eclipse.scout.rt.server.commons.authentication.PathInfoFilter.java
org.eclipse.scout.rt.server.commons.servlet.filter.authentication.SecureHttpServletRequestWrapper.java	org.eclipse.scout.rt.server.commons.authentication.SecureHttpServletRequestWrapper.java
org.eclipse.scout.rt.server.commons.authentication.ConfigFileCredentialVerifier	org.eclipse.scout.rt.platform.security.ConfigFileCredentialVerifier
org.eclipse.scout.rt.server.commons.authentication.ICredentialVerifier	org.eclipse.scout.rt.platform.security.ICredentialVerifier
org.eclipse.scout.rt.server.services.common.csv.CsvSqlSettings.java	org.eclipse.scout.rt.server.csv.CsvSettings.java
org.eclipse.scout.rt.server.services.common.jdbc.builder.AliasMapper.java	org.eclipse.scout.rt.server.jdbc.builder.AliasMapper.java
org.eclipse.scout.rt.server.services.common.jdbc.builder.DataModelEntityPartDefinition.java	org.eclipse.scout.rt.server.jdbc.builder.DataModelEntityPartDefinition.java
org.eclipse.scout.rt.server.services.common.jdbc.builder.EntityContribution.java	org.eclipse.scout.rt.server.jdbc.builder.EntityContribution.java
org.eclipse.scout.rt.server.services.common.jdbc.builder.EntityContributionUtility.java	org.eclipse.scout.rt.server.jdbc.builder.EntityContributionUtility.java

5.1.x	5.2.x/6.0.x
org.eclipse.scout.rt.server.services.common.jdbc.builder.FormDataStatementBuilder.java	org.eclipse.scout.rt.server.jdbc.builder.FormDataStatementBuilder.java
org.eclipse.scout.rt.server.services.common.jdbc.builder.FormDataStatementBuilderCheck.java	org.eclipse.scout.rt.server.jdbc.builder.FormDataStatementBuilderCheck.java
org.eclipse.scout.rt.server.services.common.jdbc.builder.TokenBasedStatementBuilder.java	org.eclipse.scout.rt.server.jdbc.builder.TokenBasedStatementBuilder.java
org.eclipse.scout.rt.server.services.common.jdbc.derby.DerbySqlStyle.java	org.eclipse.scout.rt.server.jdbc.derby.DerbySqlStyle.java
org.eclipse.scout.rt.server.services.common.jdbc.fixture.ConnectionMock.java	org.eclipse.scout.rt.server.jdbc.fixture.ConnectionMock.java
org.eclipse.scout.rt.server.services.common.jdbc.fixture.PreparedStatementMock.java	org.eclipse.scout.rt.server.jdbc.fixture.PreparedStatementMock.java
org.eclipse.scout.rt.server.services.common.jdbc.fixture.ResultSetMetaDataMock.java	org.eclipse.scout.rt.server.jdbc.fixture.ResultSetMetaDataMock.java
org.eclipse.scout.rt.server.services.common.jdbc.fixture.ResultSetMock.java	org.eclipse.scout.rt.server.jdbc.fixture.ResultSetMock.java
org.eclipse.scout.rt.server.services.common.jdbc.fixture.TableFieldBeanData.java	org.eclipse.scout.rt.server.jdbc.fixture.TableFieldBeanData.java
org.eclipse.scout.rt.server.services.common.jdbc.fixture.TableFieldData.java	org.eclipse.scout.rt.server.jdbc.fixture.TableFieldData.java
org.eclipse.scout.rt.server.services.common.jdbc.internal.legacy.LegacyStatementBuilder.java	org.eclipse.scout.rt.server.jdbc.internal.legacy.LegacyStatementBuilder.java
org.eclipse.scout.rt.server.services.common.jdbc.oracle.OracleSqlStyle.java	org.eclipse.scout.rt.server.jdbc.oracle.OracleSqlStyle.java
org.eclipse.scout.rt.server.services.common.jdbc.SqlBind.java	org.eclipse.scout.rt.server.jdbc.SqlBind.java
org.eclipse.scout.rt.server.services.common.jdbc.style.AbstractSqlStyle.java	org.eclipse.scout.rt.server.jdbc.style.AbstractSqlStyle.java
org.eclipse.scout.rt.testing.commons.ScoutAssert.java	org.eclipse.scout.rt.testing.platform.util.ScoutAssert.java

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