Eclipse Scout
Migration Guide

Scout Team

Version 6.1

Table of Contents

Text Provider Service

The method `AbstractDynamicNlsTextProviderService#getDynamicNlsBaseName` has been made public. Adjust the method in your text provider service accordingly.

Content Security Policy (CSP)

TODO IMO: Describe CSP, its implications on Scout applications and how to configure it.

By default, inline `<script>` tags in HTML files are prohibited by CSP rules. Bootstrapping JavaScript code was therefore moved to dedicated **.js* files in the `WebContent/res` folder. Existing projects using CSP have to manually perform the following steps:

1. Open each **.html* file in `your.project.ui.html/src/main/resources/WebContent` folder and check if there are any inline script parts. Only `<script>` tags with embedded JavaScript code are considered "inline". Tags with a `src` attribute don't need to be changed.
2. Transfer the content of each script part to a **.js* file in the `res` subdirectory (e.g. *index.html* ⇒ *res/index.js*) and delete the now empty <script> part.
3. Add a reference to the **.js* file in the `<head>` section using the `<scout:script>` tag, e.g.: `<scout:script src="res/index.js" />`
4. If the extracted **.js* file contains `<scout:message>` tags, they have to be moved back to the `<body>` of the corresponding **.html* file (because the NLS translation can only process HTML files). The attribute `style` has to be changed from `javascript` to `tag`.

Example:

Listing 1. login.html before migration (Scout 6.0)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Contacts Application</title>
    <scout:include template="head.html" />
    <scout:stylesheet src="res/scout-login-module.css" />
    <scout:script src="res/jquery-all-macro.js" />
    <scout:script src="res/scout-login-module.js" />
    <script> ①
      $(document).ready(function() {
        scout.login.init({texts: <scout:message style="javascript" key="ui.Login" key
="ui.LoginFailed" key="ui.User" key="ui.Password" /> });
      });
    </script>
  </head>
  <body>
    <scout:include template="no-script.html" />
  </body>
</html>
```

① Prohibited inline script.

Listing 2. login.html after migration (Scout 6.1)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Contacts Application</title>
    <scout:include template="head.html" />
    <scout:stylesheet src="res/scout-login-module.css" />
    <scout:script src="res/jquery-all-macro.js" />
    <scout:script src="res/scout-login-module.js" />
    <scout:script src="res/login.js" /> ①
  </head>
  <body>
    <scout:include template="no-script.html" />
    <scout:message style="tag" key="ui.Login" key="ui.LoginFailed" key="ui.User" key=
"ui.Password" /> ②
  </body>
</html>
```

① External script reference allowed by CSP.

② Moved from JavaScript call to `<body>`, changed style to `tag`.

Listing 3. res/login.js after migration (Scout 6.1)

```
$(document).ready(function() {
  scout.login.init(); ①
});
```

① Translated texts are extracted automatically from DOM.

Customizing CSP directives

The method `org.eclipse.scout.rt.server.commons.servlet.HttpServletControl.getCspDirectives()` is no longer available. CSP directives are now configured by the the bean `org.eclipse.scout.rt.server.commons.servlet.ContentSecurityPolicy`. To customize the rules, replace this bean with your own implementation and override the method `initDirectives()`. The bean provides fluent-style with_XXX_() methods.

By default, the `report-uri` for CSP violations is now called **/csp-report** (instead of /csp.cgi).

scout.graphics.prefSize() [JS]

The signature of JavaScript method `scout.graphics.prefSize()` has changed:

- **Old**: scout.graphics.prefSize($elem, includeMargin, options)
- **New**: scout.graphics.prefSize($elem, options)

The argument *includeMargin* was moved to the options object. See code documentation for a description of all options.

scout.ModelAdapter._send() [JS]

The signature of JavaScript method `scout.ModelAdapter._send()` has changed:

- **Old**: scout.ModelAdapter._send(type, data, delay, coalesceFunc, noBusyIndicator)
- **New**: scout.ModelAdapter._send(type, data, options)

Instead of passing individual arguments, pass all but the first two arguments in an options object: * `delay` * `coalesce` * `showBusyIndicator`

Old:

```
this._send('selected', eventData, null, function() { ... });
```

New:

```
this._send('selected', eventData, {
  coalesce: function() { ... }
});
```

Mnemonics

Mnemonics are not supported anymore. Remove all mnemonics (&) from your text files as they will not be considered anymore! && was used escape the mnemonic behaviour and display a single '&' in a text. Replace && with &!

The following methods were or will be removed:

- `StringUtility.removeMnemonic`
- `StringUtility.getMnemonic`
- `IAction.PROP_TEXT_WITH_MNEMONIC`
- `IAction.PROP_MNEMONIC`
- `IAction.getTextWithMnemonic`
- `IAction.getMnemonic`
- `strings.removeMnemonic`
- `strings.removeAmpersand`

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