Adding more scopes

Table of contents

1. Introduction

In this page you can find how to add new scopes to your web application through the use of Scopes framework classes.

2. Implement ScopeManager

First of all you have to implement `ScopeManager`, in particular, implement the `createContext`, that creates a context, for a request. You can implement the `Context` interface or use the default implementation `ContextImpl`.

3. Extend ModifierNodeVisitor

If you need to rewrite the resulting HTML page to provide some functionality for the new scope management, such as adding JavaScript code, you need to extend `ModifierNodeVisitor`. This class enables you to visit all of HTML nodes and to rewrite tag contents, or to write something before of after a tag. For a real implementation, see the class `WindowNameNodeVisitor`.

4. Implement RedirectResponseRewriter

If you want to preserve the scope functionality also when you send a "redirect" instead of a "forward", you need to implement the `RedirectResponseRewriter` interface. With such a class, you can rewrite an original URL for adding, for instance, some other parameters. See `WindowRedirectResponseRewriter` for a real example.

5. Configuration

After implementing all of the stuff above, you need to configure them in your web application. After configuring the ScopesFilter, you need to add some code to "web.xml" file (ellipsis, "...", means that there can be some other scope names, classes, etc.):

```
<init-param>
  <param-name>manager-scope-names</param-name>
  <param-value>...,myscope,...</param-value>
</init-param>
<init-param>
  <param-name>manager-class-names</param-name>
  <param-value>
    ...,
    my.scope.manager.class.ImplOfScopeManager,
    ...
  </param-value>
```

```
</init-param>
<init-param>
  <param-name>rewriter-node-visitor-classes</param-name>
  <param-value>
    ...,
    my.node.visitor.class.ImplOfModifierNodeVisitor,
    ...
  </param-value>
</init-param>
<init-param>
  <param-name>redirect-classes</param-name>
  <param-value>
    ...,
    my.redirect.request.rewriter.ImplOfRedirectRequestRewriter,
    ...
  </param-value>
</init-param>
```

6. Advanced configuration

If the normal parameter-based configuration is not enough for your needs, then you can override the default Scope configurator classes.

If you want to modify the way `State` objects are built, you can implement `StateBuilder` interface and configure it in "web.xml" this way:

```
<init-param>
  <param-name>stateBuilderClassName</param-name>
  <param-value>my.state.builder.class.ImplOfStateBuilder</param-value>
</init-param>
```

If you want to modify the way the rewriter visitors are configured, you can implement the `RewriterNodeVisitorConfigurator` interface and configure it in "web.xml" this way:

```
<init-param>
  <param-name>rewriterConfigurator</param-name>
<param-value>my.rewriter.node.visitor.configurator.class.ImplOfRewriterNodeVisitorConfi
</init-param>
```

Finally, if you want to modify the way the redirect response rewriter are configure, you can implement the `RedirectResponseRewriterConfigurator` interface and configure it in "web.xml" this way:

```
<init-param>
  <param-name>redirectConfigurator</param-name>
<param-value>my.redirect.response.rewriter.configurator.class.ImplOfRedirectResponseRew
</init-param>
```

7. Using already provided additional scopes

In this version of Scopes support for:

- init-based application scope,
- click scope,
- window scope

is provided.