Tutorial:
Securing preview

version: 1.0
Author: David Haasler

Table of contents

1 Securing page rendering..... 2
1.1 Integrating namespace..... 2
1.2 Securing..... 2
2 Securing binary methods..... 4
2.1 Integrating namespace..... 4
2.2 Securing..... 4

Requirements

Description

This tutorial describes how you can limit access to a page and a binary method.

Signs and symbols

Boxes marked with an arrow symbol and a green border contains instruction of what to do next.

This kind of boxes contains tips and tricks.

```
Source code is shown in blue boxes.
```

1 Securing page rendering

You can secure the rendering of a literal method in such a way that only users logged on (to the RenderEngine), who have particular authorization for access to the data object, can see it.

1.1 Integrating namespace

In order to move around in the context of a user within the rendering, the extension “Personalization” is necessary.

Integrate the namespace “http://onionworks.net/2006/data/personalization” into your method. The appropriate methods and elements can now be reached via the prefix “user”.

1.2 Securing

For the actual securing, insert the following, preferably directly at the beginning of the literal method:

```
<user:accessControl
  validations="read"
  notAuthenticatedUri="{$loginHref}"
  successfulAuthenticationUri="{$currentHref}"
  notAuthorizedUri="{$registerHref}"
/>
```

Parameters

Parameters	Meaning	Optional	Default behaviour if not specified
target	object on which the authorizations are to be checked.	Yes	contains value of the current object
validations	Required authorizations. Possible values: - read - modify - delete Several values separated by blanks.	Yes	read

notAuthenticatedUri	Link to a page to which forwarding is to take place if no user is currently logged in. This is usually the login page.	Yes	HTTP exception with status code 401 "Not Authenticated"
successfulAuthenticationUri	Link to a page to which forwarding is to take place if a user has successfully logged in. This is usually a link to the page which the user is trying to call.	Yes	-
notAuthorizedUri	Link to a page to which forwarding is to take place if the logged-in user does not have the necessary rights.	Yes	HTTP exception with status code "Not Authorized"

How the control works

First it is checked whether a user is logged on to the RenderEngine. If this is not the case, forwarding is done to the address in the parameter “notAuthenticatedUri”. In doing so, the HTTP GET parameter“redirect” is attached with the value “successfulAuthenticationUri”. Forwarding is therefore not automatic! After logging in successfully, forwarding can take place to the “successfulAuthenticationUri” in the login method.
If no value is specified for the parameter “notAuthenticatedUri”, an HTTP exception with status code 401 "User not authenticated" is triggered.

If a user is logged in however, it is checked whether he has the authorizations specified in “validations” for the appropriate object (“target”).
The indication of several authorizations as the parameter “validations” means a UND linking of authorizations. The check is therefore only successful if all specified authorizations are present.
If the parameter “validations” is not present, it is merely checked whether a user is logged in.

If the check of the specified authorizations fails, the user is forwarded to the page specified in the parameter “notAuthorizedUri”. If the parameter is not specified, an HTTP exception with the status code 401 “Not Authorized” is triggered.

2 Securing binary methods

If downloads are created in a secured area, these are not yet secured automatically themselves. If someone forwards such a download link, the file can also be downloaded without logging in.

In order to prevent this, the binary method must also be secured.

2.1 Integrating namespace

Integrating the namespace “Personalization” is also necessary for securing the binary methods.

2.2 Securing

A binary method is secured where the actual HTTP response is created. Instead of a <b:webResponse> element you therefore use <user:secureWebResponse>. In addition to the parameters you also set for the "normal" <b:webResponse>, you can also add the following for the secured one:

```
<user:secureWebResponse
  ...
  target="{c.id()}"
  notAuthenticatedUri="{$loginHref}"
  successfulAuthenticationUri="{$currentHref}"
/>
```

Parameters

Parameters	Meaning	Optional	Default behaviour if not specified
target	object on which the authorizations are to be checked.	Yes	contains value of the current object
notAuthenticatedUri	Link to a page to which forwarding is to take place if no user is currently logged in. This is usually the login page.	Yes	HTTP exception with status code 401 "User not Authenticated"

successfulAuthenticationUri	Link to a method to which forwarding is to take place if a user has successfully logged in. This is usually a link to the page which the user is trying to call.	Yes	-

How the control works

First it is checked whether a user is logged on to the RenderEngine. If this is not the case, forwarding is done to the address in the parameter “notAuthenticatedUri”. In doing so, the HTTP GET parameter“redirect” is attached with the value “successfulAuthenticationUri” if available. If no value is specified for the parameter “notAuthenticatedUri”, an HTTP exception with status code 401 “User not authenticated” is triggered.

If a user is logged in however, the parameter “target” is checked. If it contains a complete XLink it is checked whether the logged-in user has read authorization for it. If “target” just contains the value “onion” however (i.e. the prefix of the relevant data source), it is sufficient for a user to be logged in.

If the authorizations check fails, the user is forwarded to the page specified in the parameter “notAuthorizedUri”. If the parameter is not specified, an HTTP exception with the status code 500 “User not authorized” is triggered.