Authorizers

by Andreas Hartmann

1. Introduction

An *Authorizer* checks if an *Identity* is authorized to invoke a certain request.

The *DelegatingAuthorizerAction* tries to resolve an *AccessController* for the URL. If an *AccessController* could be resolved, its `authorize(Request)` method is used to authorize the request. If no `AccessController` could be found, the access to the request is granted for free.

The *DefaultAccessController* delegates the authorization to its *Authorizer*s. Only when all *Authorizer*s return `true`, the request is authorized.

2. PolicyAuthorizer

A *PolicyAuthorizer* uses *Policies* for authorizing. It returnes `true`, when the current *Identity* has at least one *Role* for the requested URL.

3. UsecaseAuthorizer

This *Authorizer* looks for the `lenya.usecase` request parameter and checks the usecase policy file for the *Role*s that are allowed to execute this usecase. The location of this file is defined using the `configuration` parameter which points to a URL:

```
<authorizer type="usecase">
  <parameter name="configuration"
      value="context:///lenya/pubs/mypub/config/ac/usecase-policies.xml"/>
</authorizer>
```

The usecase policy file might look as follows:

```
<?xml version="1.0"?>
<usecases xmlns="http://apache.org/cocoon/lenya/ac/1.0">
  <usecase id="create">
    <role id="editor"/>
  </usecase>
  <usecase id="rename">
    <role id="editor"/>
  </usecase>
</usecases>
```

4. WorkflowAuthorizer

The *WorkflowAuthorizer* is responsible for protecting workflow transitions. Therefore it

- looks for the `lenya.event` request parameter,
- determines the current state of the workflow instance, and
- checks if the event may be invoked by one of the current *Role*s in this state.

The *WorkflowAuthorizer* has no configuration options:

```
<authorizer type="workflow"/>
```