AclBundle Documentation

Release 1.0

Daniel Tschinder (danez)

October 15, 2015

Contents

CHAPTER 1

Requirements

The AclBundle has only a few dependencies that need to be fulfilled. It requires PHP version 5.3, as namespaces are used and the following composer packages:

- doctrine/common with at least version 2.2
- symfony/security with at least version 2.3

Note: It might also work with earlier versions of these packages, but was not tested with them. If you get it working with earlier versions, feel free to open a pull request.

CHAPTER 2

Installation

Installation is done in two steps like any other Bundle.

2.1 1. Download the bundle

The recommended way to install the AclBundle is through Composer. Just run the following command to get the latest version:

```
$ composer require project-a/acl-bundle
```

2.2 2. Enable the bundle

Enable the bundle in the kernel of your application:

```
// app/AppKernel.php

public function registerBundles()
{
    $bundles = array(
        // ...
        new ProjectA\Bundle\AclBundle\ProjectAAclBundle(),
    );
}
```

Note: After enabling the Bundle in your kernel you will not notice any difference, as using the provided services is optional and the eventlistener for doctrine is not enabled by default.

This was already everything you need to do to get started with the AclBundle.

CHAPTER 3

Guides

Learn more about how to use this bundle.

3.1 Retrieving Service Classes

The AclBundle registeres 3 service classes in the DependencyInjection Container, the `AclManager`, the `ClassAceManager` and the `ObjectAceManager`.

3.1.1 Acl Manager

To retrieve the AclManager simply do this:

```
$aclManager = $container->get('projecta_acl.manager');
```

The AclManager has 2 public methods to retrieve the Ace Managers:

manageObjectAces() returns the ObjectAceManager

manageClassAces() returns the ClassAceManager

3.1.2 Class/Object Ace Manager

To get one of the ACE Managers you can either use the methods `manageObjectAces()` and `manageClassAces()` (see above) or retrieve them directly from the DIC. Whatever suits your needs best.

```
$objectAceManager = $container->get('projecta_acl.ace.objectmanager');
$classAceManager = $container->get('projecta_acl.ace.classmanager');
```

3.2 Basic Usage

Note: Notice that the two AceManagers have a fluid interface. You can chain calls like this:

```
$aceManager->grant(...)->grant(...)->revoke(...);
```

3.2.1 Granting permissions

Object

You can grant rights on objects with one simple call to the `grant()` method.

```
$aceManager->grant($object, MaskBuilder::MASK_OWNER, $user);
```

This example marks `$user` as owner of exactly this instance `$object`.

Class

Granting rights for classes works the same as on objects. When calling `grant()` on the `ClassAceManager` it will grant `$user` rights on all instances from the class of `$object`.

Field

Granting rights to only a property of an object or class can also be done with the same method. The only difference is to supply the fourth parameter `$field`.

```
$aceManager->grant($object, MaskBuilder::MASK_OWNER, $user, 'myProperty');
```

3.2.2 Revoking permissions

Object

Removing previously granted rights can be done by calling the `revoke()` method.

```
$aceManager->revoke($object, MaskBuilder::MASK_OWNER, $user);
```

This example removes `$user` as owner of exactly this instance `$object`.

If you want to remove all granted rights on the object for one user, there is a special method `revokeAllForIdentity()`.

```
$aceManager->revokeAllForIdentity($object, $user);
```

Class

Revoking rights for classes works the same as on objects. When calling `revoke()` or `revokeAllForIdentity()` on the `ClassAceManager` it will revoke the previously granted rights for `$user` on all instances from the class of `$object`.

Field

If you want to revoke the granted rights for a field, you just need to supply the 4th (`revoke()`) or 3rd (`revokeAllForIdentity`) argument, which specifies the name of the field.

```
$aceManager->revoke($object, MaskBuilder::MASK_OWNER, $user, 'myfield');
$aceManager->revokeAllForIdentity($object, $user, 'field');
```

3.2.3 Checking permissions

If you want to check if the current user is granted a permission you can do that by calling `isGranted()`.

```
if ($aclManager->isGranted(MaskBuilder::MASK_EDIT, $object)) {
    // editing $object is allowed
}

if ($aclManager->isGranted(MaskBuilder::MASK_EDIT, $object, 'myfield')) {
    // editing $object->myfield is allowed
}
```

3.2.4 Deleting ACLs

Deleting the ACL for an `$object` is the same as if you would remove all entries from the storage. This function is probably very useful when you are going to remove the `$object` and want to cleanup all it's ACL entries.

```
$aceManager->deleteAcl($object);
```

Note: If you want to automate the deletion of all acl entries for an domain object when it gets deleted have a look at Doctrine Cleanup Listener

3.3 Doctrine Cleanup Listener

3.4 Configuration Reference

These are all available configuration options for this bundle with their default values.

```
projecta_acl:
    remove_orphans: false|true (default: false)
    default_strategy: all|any|equal (default: all)
```

3.5 Changelog

3.5.1 1.0.1

Released on 2015-03-15

- AclManager::preload() will throw exceptions if Acl is not found
- Fixed all deprecate usages for symfony >=2.6
- Documentation updates

3.5.2 1.0.0

Released on 2014-11-21

- Initial Release