SensioLabs

Symfony

Hacking & Extending Symfony2
SF2C3

Creating a compiler pass

Goal

We want to be able to automatically register new dictionary file loaders into the word list service from another bundle.

Requirements

Adding a new **compiler pass** that processes **tagged services** and registers these services directly into the container.

What is a compiler pass?

A compiler pass allows to execute some code to modify the service container configuration just before it's compiled down to PHP code.

Examples of native compiler passes

- Resolving dynamic container parameters
- Registering Twig extensions
- Registering Doctrine event listeners
- Removing orphan services
- Optimizing the service container

Tagging services

```
<service
    id="sensio_hangman.loader.txt"
    class="%sensio_hangman.loader.txt.class%"
    public="false">
    <tag name="hangman.dictionary_loader" type="txt"/>
</service>

<service
    id="sensio_hangman.loader.xml"
    class="%sensio_hangman.loader.xml.class%"
    public="false">
    <tag name="hangman.dictionary_loader" type="xml"/>
</service>
```

Creating a compiler pass class

```
namespace Sensio\Bundle\HangmanBundle\DependencyInjection\Compiler;

use Symfony\Component\DependencyInjection\Compiler\CompilerPassInterface;
use Symfony\Component\DependencyInjection\ContainerBuilder;

class DictionaryLoaderCompilerPass implements CompilerPassInterface
{
    public function process(ContainerBuilder $container)
    {
        // ...
    }
}
```

```
// ...
use Symfony\Component\DependencyInjection\Reference;

class DictionaryLoaderCompilerPass implements CompilerPassInterface
{
  public function process(ContainerBuilder $container)
  {
    if (!$container->hasDefinition('sensio_hangman.word_list')) {
      return;
    }

    $wordList = $container->getDefinition('sensio_hangman.word_list');
    $ids = $container->findTaggedServiceIds('hangman.dictionary_loader');

    foreach ($ids as $id => $attributes) {
        $type = $attributes[0]['type'];
        $wordList->addMethodCall('addLoader', array($type, new Reference($id)));
    }

    $calls = $wordList->getMethodCalls();
    $calls = array_reverse($calls);

    $wordList->setMethodCalls($calls);
  }
}
```

Registering the compiler pass

```
namespace Sensio\Bundle\HangmanBundle;

use Sensio\[…]\DependencyInjection\Compiler\DictionaryLoaderCompilerPass;
use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\HttpKernel\Bundle\Bundle;

class SensioHangmanBundle extends Bundle
{
    public function build(ContainerBuilder $container)
    {
        $container->addCompilerPass(new DictionaryLoaderCompilerPass());
    }
}
```

Creating a custom voter

Goal

We want to allow users playing the game between 8:00 am and 7:00 pm. Access to the game will be forbidden otherwise.

Requirements

The schedule range and the timezone must be configurable.

The Voter Class

```
namespace Sensio\Bundle\HangmanBundle\Security\Voter;

use Symfony\Component\Security\Core\Authorization\Voter\VoterInterface;
use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;

class ScheduleVoter implements VoterInterface
{
    private $minHours;
    private $maxHours;
    private $timezone;

    public function __construct($minHours, $maxHours, $timezone)
    {
        $this->minHours = $minHours;
        $this->maxHours = $maxHours;
        $this->timezone = new \DateTimeZone($timezone);
    }
}
```

```
namespace Sensio\Bundle\HangmanBundle\Security\Voter;

class ScheduleVoter implements VoterInterface
{
    public function supportsAttribute($attribute)
    {
        return true;
    }

    public function supportsClass($class)
    {
        return true;
    }
}
```

```
namespace Sensio\Bundle\HangmanBundle\Security\Voter;

class ScheduleVoter implements VoterInterface
{
    function vote(TokenInterface $token, $object, array $attributes)
    {
        $date = new \DateTime('H', $this->timezone);
        $hours = (int) $date->format('H');

        if ($hours >= $this->minHours && $hours < $this->maxHours) {
            return self::ACCESS_GRANTED;
        }

        return self::ACCESS_DENIED;
    }
}
```

Register the voter as a service

```
<?xml version="1.0" ?>
<container ...>
  <parameters>
    <parameter key="sensio_hangman.schedule_voter.class">Sensio\[…]\ScheduleVoter</parameter>
    <parameter key="sensio_hangman.schedule_voter.min_hours »>8</parameter>
    <parameter key="sensio_hangman.schedule_voter.max_hours">19</parameter>
    <parameter key="sensio_hangman.schedule_voter.timezone">Europe/Paris</parameter>
  </parameters>
  <services>
    <service id="sensio_hangman.schedule_voter"
      class="%sensio_hangman.schedule_voter.class%"
      public="false"
    >
      <argument>%sensio_hangman.schedule_voter.min_hours%</argument>
      <argument>%sensio_hangman.schedule_voter.max_hours%</argument>
      <argument>%sensio_hangman.schedule_voter.timezone%</argument>
      <tag name="security.voter"/>
    </service>
  </services>
</container>
```

Increasing the security level

By default, the application grants the access if at least one voter returns a positive vote.

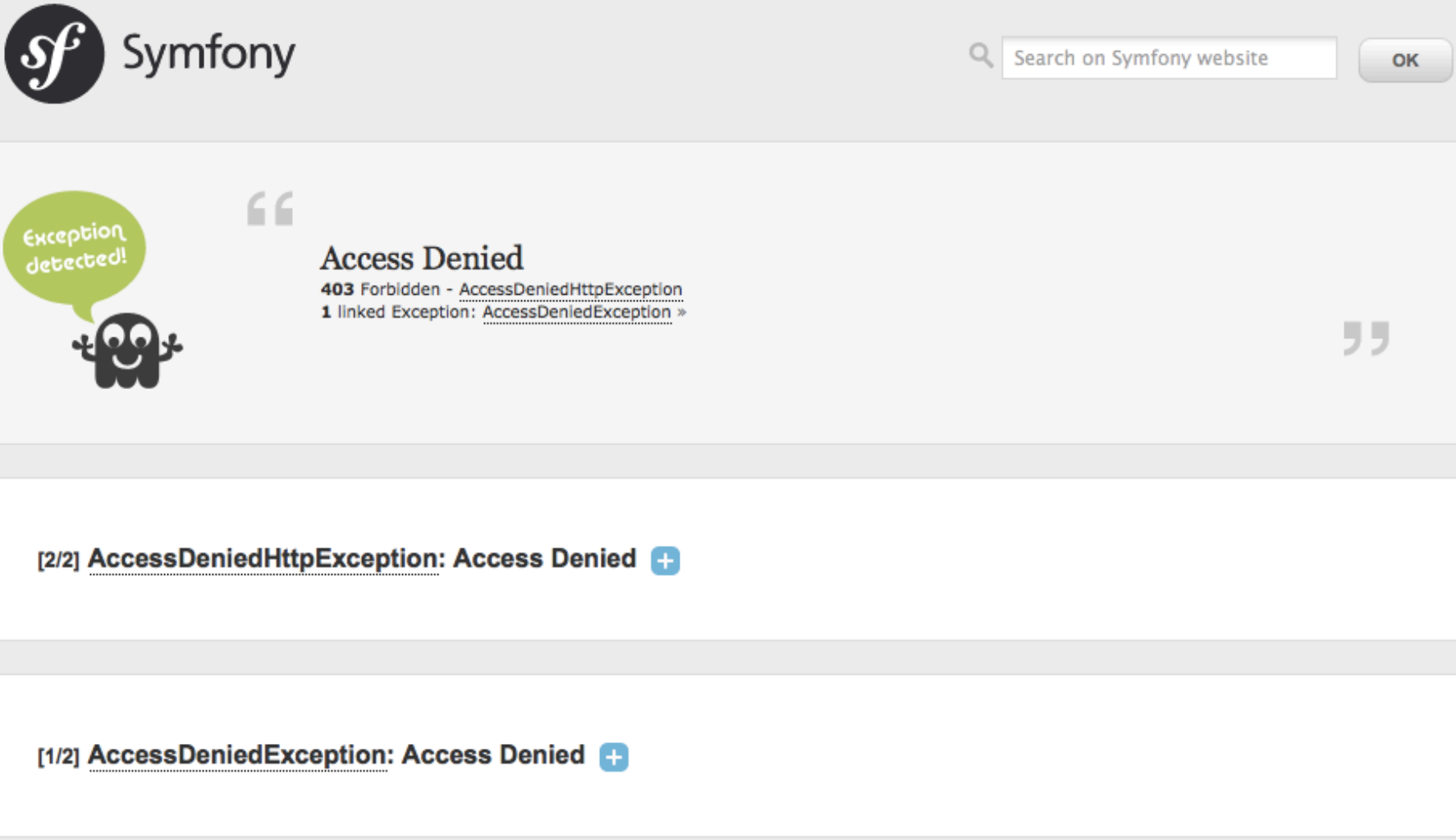
Nevertheless, the access must be denied if at least the schedule voter returns a negative decision.

Increasing the security level

```
# app/config/security.yml
security:
    access_decision_manager:
        strategy: unanimous

    # […]
```

The voter in action



Training Department

SensioLabs Training

92-98 Boulevard Victor Hugo

92 115 Clichy Cedex

FRANCE

Phone: +33 140 998 211

Email: training@sensiolabs.com

symfony.com - trainings.sensiolabs.com