How to listen to Core events

Version compatibility
This recipe is compatible with **eZ Publish 5.2 / 2013.07**

Description

When you interact with the Public API, and with the content Repository in particular, **Signals** may be sent out, allowing you to react on actions triggered by the Repository. Those signals can be received by dedicated services called **Slots**.

To learn more about SignalSlot in eZ Publish, please refer to the dedicated documentation page.

Signals reference

This recipe will describe how to register a Slot for a dedicated Signal.

Registering a Slot for a given Signal

As described in the SignalSlot documentation, a Slot is roughly like an event listener and must extend `eZ\Publish\Core\SignalSlot\Slot`.

A typical implementation is the following:

OnPublishSlot

```
namespace Acme\TestBundle\Slot;

use eZ\Publish\Core\SignalSlot\Slot as BaseSlot;
use eZ\Publish\API\Repository\Repository;
use eZ\Publish\SignalSlot\Signal;

class OnPublishSlot extends BaseSlot
{
    /**
     * @var \eZ\Publish\API\Repository\ContentService
     */
    private $contentService;

    public function __construct( ContentService $contentService )
    {
        $this->contentService = $contentService;
    }

    public function receive( Signal $signal )
    {
        if ( !$signal instanceof Signal\ContentService\PublishVersionSignal )
        {
            return;
        }

        // Load published content
        $content = $this->contentService->loadContent( $signal->contentId, null,
$signal->versionNo );
        // Do stuff with it...
    }
}
```

`OnPublishSlot` now needs to be registered as a service in the ServiceContainer and identified as a valid Slot:

services.yml (in your bundle)

```
parameters:
    my.onpublish_slot.class: Acme\TestBundle\Slot\OnPublishSlot

services:
    my.onpublish_slot:
        class: %my.onpublish_slot.class%
        arguments: [@ezpublish.api.service.content]
        tags:
            - { name: ezpublish.api.slot, signal: ContentService\PublishVersionSignal
}
```

Service tag **`ezpublish.api.slot`** identifies your service as a valid Slot. The signal part (mandatory) says that this slot is listening to **`ContentService\PublishVersionSignal`** (shortcut for `\eZ\Publish\Core\SignalSlot\Signal\ContentService\PublishVersionSignal`)
.

Internal signals emitted by Repository services are always relative to `eZ\Publish\Core\SignalSlot\Signal` namespace.

Hence `ContentService\PublishVersionSignal` means `eZ\Publish\Core\SignalSlot\Signal\ContentService\PublishVersionSignal`.

Tip
You can register a slot for any kind of signal by setting `signal` to **`*`** in the service tag.

Using a basic Symfony event listener

eZ Publish comes with a generic slot that converts signals (including ones defined by user code) to regular event objects and expose them via the EventDispatcher. This makes it possible to implement a simple event listener/subscriber if you're more comfortable with this approach.

All you need to do is to implement an event listener or subscriber and register it.

Simple example

This very simple example will just log the received signal.

services.yml (in your bundle)

```
parameters:
    my.signal_listener.class: Acme\TestBundle\EventListener\SignalListener

services:
    my.signal_listener:
        class: %my.signal_listener.class%
        arguments: [@logger]
        tags:
            - { name: kernel.event_subscriber }
```

```php
<?php
namespace Acme\TestBundle\EventListener;

use eZ\Publish\Core\MVC\Symfony\Event\SignalEvent;
use eZ\Publish\Core\MVC\Symfony\MVCEvents;
use Psr\Log\LoggerInterface;
use Symfony\Component\EventDispatcher\EventSubscriberInterface;

class SignalListener implements EventSubscriberInterface
{
    /**
     * @var \Psr\Log\LoggerInterface
     */
    private $logger;

    public function __construct( LoggerInterface $logger )
    {
        $this->logger = $logger;
    }

    public function onAPISignal( SignalEvent $event )
    {
        $signal = $event->getSignal();
        // You may want to check the signal type here to react accordingly
        $this->logger->debug( 'Received Signal: ' . print_r( $signal, true ) );
    }

    public static function getSubscribedEvents()
    {
        return array(
            MVCEvents::API_SIGNAL => 'onAPISignal'
        );
    }
}
```