Language Switcher

Version compatibility
Language switcher in Symfony stack is compatible with **eZ Publish 5.3 / 2014.05**

Description

A content item can be translated in several languages. Those languages are configured in the system and exposed in SiteAccesses via a prioritized list of languages:

ezpublish.yml

```
ezpublish
    system:
        eng:
            languages: [eng-GB]
        # In fre siteaccess, fre-FR is always preferred, and fallback to eng-GB if
needed.
        fre:
            languages: [fre-FR, eng-GB]
```

When visiting a Content item, it may be useful to let the user switch from one translation to another, more appropriate to him. This is precisely the goal of the language switcher.

The language switcher relies on the Cross-SiteAccess linking feature to generate links to the content's translation, and on RouteReference feature .

Tip
If you install the DemoBundle with at least 2 different languages, you will be able to see the Language Switcher and to test it.

Usage

Configuration: explicit *translation SiteAccesses*

Configuration is not mandatory, but can help to distinguish which SiteAccesses can be considered *translation SiteAccesses*.

ezpublish.yml

```
ezpublish:
    siteaccess:
        default_siteaccess: eng
        list:
            - ezdemo_site
            - eng
            - fre
            - ezdemo_site_admin

        groups:
            ezdemo_frontend_group:
                - ezdemo_site
                - eng
                - fre

    # ...

    system:
        # Specifying which SiteAccesses are used for translation
        ezdemo_frontend_group:
            translation_siteaccesses: [fre, eng]
        eng:
            languages: [eng-GB]
        fre:
            languages: [fre-FR, eng-GB]
        ezdemo_site:
            languages: [eng-GB]
```

Note: Top prioritized language is always used for as the SiteAccess language reference (e.g. `fre-FR` for `fre` SiteAccess in the example above).

If several translation SiteAccesses share the same language reference, **the first declared SiteAccess always wins**.

Configuration: more complex translation setup

There are some cases where your SiteAccesses share settings (repository, content settings...), but you don't want all of them to share the same `translation_siteaccesses` setting. This can be the case when you use SiteAccesses for mobile.

Solution is as easy as defining new groups:

ezpublish.yml

```
ezpublish:
    siteaccess:
        default_siteaccess: eng
        list:
            - ezdemo_site
            - eng
            - fre
            - mobile_eng
            - mobile_fre
            - ezdemo_site_admin

        groups:
            # This group can be used for common front settings
            ezdemo_common_group:
                - ezdemo_site
                - eng
                - fre
                - mobile_eng
                - mobile_fre

            ezdemo_frontend_group:
                - ezdemo_site
                - eng
                - fre

            ezdemo_mobile_group
                - mobile_eng
                - mobile_fre

    # ...

    system:
        # Translation SiteAccesses for regular frontend
        ezdemo_frontend_group:
            translation_siteaccesses: [fre, eng]

        # Translation SiteAccesses for mobile frontend
        ezdemo_mobile_group:
            translation_siteaccesses: [mobile_fre, mobile_eng]

        eng:
            languages: [eng-GB]
        fre:
            languages: [fre-FR, eng-GB]
        ezdemo_site:
            languages: [eng-GB]

        mobile_eng:
            languages: [eng-GB]
        mobile_fre:
            languages: [fre-FR, eng-GB]
```

Configuration: using implicit *related SiteAccesses*

If `translation_siteaccesses` setting is not provided, implicit *related SiteAccesses* will be used instead. SiteAccesses are considered *related* if they share:

- The same repository
- The same root location Id (see Multisite feature)

In a template

To generate a language switch link, you need to generate the `RouteReference`, with the `language` parameter. This can easily be done with `ez_route()` Twig function:

```
{# Given that "location" variable is a valid Location object #}
<a href="{{ url( ez_route( location, {"language": "fre-FR"} ) ) }}">{{
ez_content_name( content ) }}</a>

{# Generating a link to a declared route instead of Location #}
<a href="{{ url( ez_route( 'my_route', {"language": "fre-FR"} ) ) }}">My link</a>
```

You can also omit the route, in this case, the current route will be used (i.e. switch the current page):

```
{# Using Twig named parameters #}
<a href="{{ url( ez_route( params={"language": "fre-FR"} ) ) }}">My link</a>

{# Identical to the following, using ordered parameters #}
<a href="{{ url( ez_route( null, {"language": "fre-FR"} ) ) }}">My link</a>
```

Using sub-requests

When using sub-requests, you lose the context of the master request (e.g. current route, current location...). This is because sub-requests can be displayed separately, with ESI or Hinclude.

If you want to render language switch links in a sub-request with a correct `RouteReference`, you must pass it as an argument to your sub-controller from the master request.

```
{# Render the language switch links in a sub-controller #}
{{ render( controller( 'AcmeTestBundle:Default:languages', {'routeRef': ez_route()} )
) }}
```

```
namespace Acme\TestBundle\Controller;

use eZ\Bundle\EzPublishCoreBundle\Controller;
use eZ\Publish\Core\MVC\Symfony\Routing\RouteReference;

class DefaultController extends Controller
{
    public function languagesAction( RouteReference $routeRef )
    {
        return $this->render( 'AcmeTestBundle:Default:languages.html.twig', array(
'routeRef' => $routeRef ) );
    }
}
```

```
{# languages.html.twig #}

{# Looping over all available languages to display the links #}
{% for lang in ezpublish.availableLanguages %}
    {# This time, we alter the "siteaccess" parameter directly. #}
    {# We get the right siteaccess with the help of ezpublish.translationSiteAccess()
helper #}
    {% do routeRef.set( "siteaccess", ezpublish.translationSiteAccess( lang ) ) %}
    <a href="{{ url( routeRef ) }}">{{ lang }}</a><br />
{% endfor %}
```

- `ezpublish.translationSiteAccess( language )` returns the SiteAccess name for provided language (or `null` if it cannot be found)
- `ezpublish.availableLanguages()` returns the list of available languages.

Using PHP

You can easily generate language switch links from PHP too, with the `RouteReferenceGenerator` service:

```
// Assuming we're in a controller
/** @var
\eZ\Publish\Core\MVC\Symfony\Routing\Generator\RouteReferenceGeneratorInterface
$routeRefGenerator */
$routeRefGenerator = $this->get( 'ezpublish.route_reference.generator' );
$routeRef = $routeRefGenerator->generate( $location, array( 'language' => 'fre-FR' )
);
$link = $this->generateUrl( $routeRef );
```

You can also retrieve all available languages with the `TranslationHelper`:

```
/** @var \eZ\Publish\Core\Helper\TranslationHelper $translationHelper */
$translationHelper = $this->get( 'ezpublish.translation_helper' );
$availableLanguages = $translationHelper->getAvailableLanguages();
```