RouteReference

> **Version compatibility**
> Route references are compatible with **eZ Publish 5.3 / 2014.05**

- Description
- Solution
- Usage
  - Twig
  - PHP
- Extending the RouteReference generation process

Description

Sometimes, when generating links to a resource, you need to modify the default router's behavior.

Use cases can be:

- Language switch links
- Download links
- Pass a Content item instead of a Location (and use its `mainLocationId`)

Solution

The concept of **RouteReference** has been introduced, which works in the same way of Symfony's `ControllerReference` for sub-requests. A `RouteReference` represents a route (to a location object, a declared route...) with its parameters and can be passed to the `Router` for link generation.

The advantage of a `RouteReference` is that its params can be modified later, and then passed to the router (e.g. to generate a link to the same location in several different languages).

Furthermore, the `RouteReference` generation process can be extended to fit specific needs.

Usage

Twig

Prototype:

```
ez_route( [routing_resource[, parameters_hash]] )
```

- `routing_resource` can be any valid resource (route name, Location object...). If omitted (`null`), the current route will be taken into account.
- `parameters_hash` is a hash with arbitrary key/values. It will be passed to the router in the end.

Minimal usage is pretty straightforward:

```
{# With a declared route. #}
{% set routeRef = ez_route( "my_route" ) %}

{# With a location, given "location" variable is a valid Location object. #}
{% set routeRef = ez_route( location ) %}

{# Then pass the routeRef variable to path() to generate the link #}
<a href="{{ path( routeRef ) }}">My link</a>
```

Passing parameters and play with the RouteReference:

```
{% set routeRef = ez_route( "my_route", {"some": "thing"} ) %}

{# You can then add parameters further on #}
{% do routeRef.set( "foo", ["bar", "baz"] ) %}

{# Or even modify the route resource #}
{% do routeRef.setRoute( "another_route" ) %}

<a href="{{ path( routeRef ) }}">My link</a>
```

PHP

You can easily generate links based on a `RouteReference` 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( 'my_route', array( 'some' => 'thing' );
$routeRef->set( 'foo', array( 'bar', 'baz' ) );
$routeRef->setRoute( 'another_route' );

$link = $this->generateUrl( $routeRef );
```

Extending the RouteReference generation process

When generating the route reference, the `RouteReferenceGenerator` service fires an `MVCEvents::ROUTE_REFERENCE_GENERATION` (*ezpublish.routing.reference_generation*) event. This event can be listened to in order to modify the final route reference (adding/changing parameters, changing the route name...).

All listeners receive a `eZ\Publish\Core\MVC\Symfony\Event\RouteReferenceGenerationEvent` object, which contains the current request object and the route reference.

```
namespace Acme\AcmeTestBundle\EventListener;

use eZ\Publish\Core\MVC\Symfony\Event\RouteReferenceGenerationEvent;
use eZ\Publish\Core\MVC\Symfony\MVCEvents;
use Symfony\Component\EventDispatcher\EventSubscriberInterface;

class MyRouteReferenceListener implements EventSubscriberInterface
{
    public static function getSubscribedEvents()
    {
        return array(
            MVCEvents::ROUTE_REFERENCE_GENERATION => 'onRouteReferenceGeneration'
        );
    }

    public function onRouteReferenceGeneration( RouteReferenceGenerationEvent $event )
    {
        $routeReference = $event->getRouteReference();
        $request = $event->getRequest();

        // Let's say we want to change the route name if "some_parameter" param is
present
        if ( $routeReference->has( 'some_parameter' )
        {
            $routeReference->setRoute( 'a_specific_route' );
            // We remove "some_parameter", as we don't need it any more
            $routeReference->remove( 'some_parameter' );
            // We add another parameter, just because it's fun :-)
            $routeReference->set( 'another_parameter', array( 'parameters', 'are',
'fun' );
        }
    }
}
```

Service declaration:

```
# AcmeTestBundle/Resources/config/services.yml
parameters:
    acme.my_route_reference_listener.class:
Acme\AcmeTestBundle\EventListener\MyRouteReferenceListener

services:
    acme.my_route_reference_listener:
        class: %acme.my_route_reference_listener.class%
        tags:
            - { name: kernel.event_subscriber }
```

> A *real life* implementation example can be the LanguageSwitcher (`eZ\Publish\Core\MVC\Symfony\EventListener\LanguageSwitchListener`).