Cross-siteaccess links

When using the *multisite* feature, it is sometimes useful to be able to **generate cross-links** between the different sites.
This allows you to link different resources referenced in the same content repository, but configured independently with different tree roots.

Usage

Twig example

```
 {# Linking a location #}
<a href="{{ url( 'ez_urlalias', {'locationId': 42, 'siteaccess':
'some_siteaccess_name'} ) }}">{{ ez_content_name( content ) }}</a>

{# Linking a regular route #}
<a href="{{ url( "some_route_name", {"siteaccess": "some_siteaccess_name"} ) }}">Hello
world!</a>
```

See ez_urlalias documentation page, for more information about linking to a Location

PHP example

```
namespace Acme\TestBundle\Controller;

use eZ\Bundle\EzPublishCoreBundle\Controller as BaseController;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;

class MyController extends BaseController
{
    public function fooAction()
    {
        // ...

        $location = $this->getRepository()->getLocationService()->loadLocation( 123 );
        $locationUrl = $this->generateUrl(
            $location,
            array( 'siteaccess' => 'some_siteaccess_name' ),
            UrlGeneratorInterface::ABSOLUTE_PATH
        );

        $regularRouteUrl = $this->generateUrl(
            'some_route_name',
            array( 'siteaccess' => 'some_siteaccess_name' ),
            UrlGeneratorInterface::ABSOLUTE_PATH
        );

        // ...
    }
}
```

Important
As siteaccess matchers can involve hosts and ports, it is **highly recommended** to generate cross-siteaccess links in an absolute form (e.g. using `url()` Twig helper).

Troubleshooting

- The **first matcher succeeding always wins**, so be careful when using *catch-all* matchers like `URIElement`.
- If passed siteaccess name is not a valid one, an `InvalidArgumentException` will be thrown.
- If matcher used to match the provided siteaccess doesn't implement `VersatileMatcher`, the link will be generated for the current siteaccess.
- When using `Compound\LogicalAnd`, all inner matchers **must match**. If at least one matcher doesn't implement `VersatileMatcher`, it will fail.
- When using `Compound\LogicalOr`, the first inner matcher succeeding will win.

Under the hood

To implement this feature, a new `VersatileMatcher` was added to allow siteaccess matchers to be able to *reverse-match*.
All existing matchers implement this new interface, except the Regexp based matchers which have been deprecated.

The siteaccess router has been added a `matchByName()` method to reflect this addition. Abstract URLGenerator and `DefaultRouter` have been updated as well.

Note
Siteaccess router public methods have also been extracted to a new interface, `SiteAccessRouterInterface`.

Related topics:

- Siteaccess