

Binary files URL handling

5.4 / 2014.09

IO URL decoration

By default, images and binary files referenced by content will be served from the same server than the application, like `/var/ezdemo_site/storage/images/3/6/4/6/6463-1-eng-GB/kidding.png`. This is the default semantic configuration:

```
ezpublish:
  system:
    default:
      io:
        url_prefix: "$var_dir$/$storage_dir$"
```

`var_dir` and `$storage_dir$` are dynamic, [siteaccess aware settings](#), and will be replaced by those settings value in the execution context.

URL decorators are an eZ Publish 5 features. If an image field is displayed via a legacy callback or legacy template, no decoration will be applied.

Using a static server for images

One common use-case is to use an optimized nginx to serve images in an optimized way. The example image above could be made available as `http://static.example.com/images/3/6/4/6/6463-1-eng-GB/kidding.png`, by setting up a server that uses `ezpublish/ezpublish_legacy/var/ezdemo_site/storage`. The eZ Publish configuration would be as follows:

```
ezpublish:
  system:
    default:
      io:
        url_prefix: "http://static.example.com/"
```

Legacy compatibility

Legacy still requires non absolute path to store images (`var/site/storage/images...`). In order to work around this, an `UrlRedecorator`, that converts back and forth between the legacy uri prefix and the one in use in the application, has been added. It is used in all places where a legacy URL is returned/expected, and takes care of making sure the value is as expected.

Internals

Any `BinaryFile` returned by the public API is prefixed with the value of this setting, internally stored as `ezsettings.scope.io.url_prefix`.

Dynamic container settings

Those settings are siteaccess aware.

`io.url_prefix`

Default value: `$var_dir/$storage_dir`

Example: `/var/ezdemo_site/storage`

Used to configure the default URL decorator service (`ezpublish.core.io.default_url_decorator`, used by all binarydata handlers to generate the URI of loaded files. It is always interpreted as an absolute URI, meaning that unless it contains a scheme (`http://`, `ftp://`), it will be prepended with a `'/`.

`io.legacy_url_prefix`

Default value: `$var_dir/$storage_dir`

Example: `var/ezdemo_site/storage`

Used by the legacy storage engine to convert images public URI to a format it understands. Unlike `io.url_prefix`, it is not an absolute link. Can not be overridden using semantic configuration. Changing this value will break compatibility for the legacy backoffice.

`io.root_dir`

Example: `%ezpublish_legacy.root_dir%/$var_dir/$storage_dir`

Default value: `/var/www/ezpublish/ezpublish_legacy/var/ezdemo_site/storage`

Physical path where binary files are stored on disk. Can not be overridden using semantic configuration. Changing this value will break compatibility for the legacy backoffice.

Services

url decorators

An `UrlDecorator` decorates and undecorates a given string (`url`) in some way. It has two mirror methods: `decorate` and `undecorate`.

Two implementations are provided: `Prefix`, and `AbsolutePrefix`. They both add a prefix to an URL, but `AbsolutePrefix` will ensure that unless the prefix is an external URL, the result will be prepended with `/`.

Three `UrlDecorator` services are introduced:

- `ezpublish.core.io.prefix_url_decorator` Used by the binarydata handlers to decorate all uris sent out by the API. Uses `AbsolutePrefix`.
- `ezpublish.core.io.image_fieldtype.legacy_url_decorator` Used via the `UrlRedecorator` (see below) by various legacy elements (`Converter`, `Storage Gateway`...) to generate its internal storage format for uris. Uses a `Prefix`, not an `AbsolutePrefix`, meaning that no leading `/` is added.

In addition, an `UrlRedecorator` service, `ezpublish.core.io.image_fieldtype.legacy_url_redecorator`, uses both decorators above to convert URIs between what is used on the new stack, and what format legacy expects (relative urls from the `ezpublish` root).