Tutorial:
Readable URLs - Adjusting the literal URI-format

version: 1.0
Author: David Haasler

Table of contents

Requirements

The developer must have write access to the web.config file of the Render engine where appropriate, or at least read access.

It is also assumed that a functional website is already available.

Description

In this tutorial it is explained how to create readable URLs for your website using the transformations. It is assumed here that no methods are available for this in the project.

Signs and symbols

Boxes marked with an arrow symbol and a green border contains instruction of what to do next.

This kind of boxes contains tips and tricks.

```
Source code is shown in blue boxes.
```

1 Introduction

After drawing up and conducting a new project from scratch, URLs usually have the following format:

```
www.contoso.de/language=de/274
```

The part “language=de” represents here all possible parameters which are needed for the rendering of a page. “274” is the ID of the object that is currently being rendered.
In the example above, the method “default ()” is implicitly called on the object with the ID “274”. And with the parameter “language “, which is transferred the value “de”.

If, instead of the “default()”method, a different one is called, the method name with a tilde in front is also listed in the URL, in the following way for example: “www.contoso.de/language=de/~default.search/274”.

The goal is to make these URLs readable. Here we always differentiate between two different approaches: readable systemic URLs and short URLs.

Readable systemic URLs still contain the object ID, parameters and, where appropriate, the method name. They can however be dynamically extended to include a readable name of the current page as well as the names of the “parent pages”, so as to show the click path in the URL also. The example above could then look roughly as follows:

```
www.contoso.de/ueber-uns/standorte/language=de/274/dortmund
```

The presence of the parameters, ID and, if necessary, the method, leads to the web page still being callable even without the readable components. The URL can be resolved correctly, even in the case of typing errors in these parts.

Short URLs are intended for use with landing pages. They are meant to represent a URL in as short a manner as possible so as to be able to communicate links to subpages or publish them in such a way that users are able to remember the address easily on the one hand and also type them in manually. Short URLs can be made with the Quicklinks-Modul.

The example above could then look roughly as follows:
www.contoso.de/dortmund

Both variants have their pros and cons. It should therefore be well considered which variant is to be used for which purposes. The following table gives a short overview. A combination of the two variants usually makes good sense.

	Advantage	Disadvantage
Readable systemic URLs	› Usable for search engines (SEO) › Dynamic generation, whereby maintaining the path elements is possible on each visitable object › The creation and resolution of URLs is handled in the Render Engine and is still done via the ID. › Changing the path elements is relatively unproblematic. › Performance optimisation	› Not usable for landing pages due to ID and parameters
Short URLs	› Usable for landing pages › Usable for search engines (SEO)	› The Application Request Routing (ARR) of the IIS is used for the short URLs. › The generation and resolution is not dynamic, but equates to a “reference work”. If there is a corresponding number of quicklinks, this can slow down the performance of the page considerably.

- Maintenance of path elements not directly on the object but somewhere else (by referencing the visitable object)

This tutorial describes the ways of creating readable systemic URLs.

2 A look at the web.config

As previously mentioned, the second example URL from the introduction can be created using transformations. Since the Render Engine is used for this however, it needs information as to where it needs to look.

In the web.config file of the Render Engine, there is a section <uriFormat> below “//onion/renderengine”. This contains all necessary information for the two directions needed for a URI format: the creation (<builder>) and the resolution (<parser>).

In the case of a standard installation, the section looks roughly as follows.

```
<uriFormat staticFormatterPath="/contoso/preview/binary.ashx/data"
staticParserPath="/contoso/preview/binary.ashx/data">
  <builder type="Onion.RenderEngine.DataSource.OnionUriPathFormat,
Onion.RenderEngine.DataSource">
    <format
        literal-path="/contoso/preview/page.ashx"
        binary-path="/contoso/preview/binary.ashx"
        lookup-method="uri"
        lookup-method-parameter="language"
        dispatcher-method="root-dispatcher"
        binary-lookup-method="binary-uri"
        binary-lookup-method-parameter="select"
        quicklinks-method="none"
        force-lower-case="true"
        invalid-character="-"
    />
  </builder>
  <parser type="Onion.RenderEngine.DataSource.OnionUriPathFormat,
Onion.RenderEngine.DataSource">
    <format
        dispatcher-method="root-dispatcher"
        quicklinks-method="none"
        force-lower-case="true"
        invalid-character="-"
    />
  </parser>
</uriFormat>
```

The two attributes »lookup-method« and, if necessary, »lookup-method-parameter« are important for influencing the URI format using transformations. The method “uri” is configured here with the parameter “language” as standard.

In addition to this explicit parameter, the method name is also always transferred into the method as the parameter “target-method”.

This means that the Render Engine, when creating a URL, looks for an XML method with the name “uri” for the relevant object (Bezüge im Satz richtig verstanden?) on which the call is made, and passes through the parameters “target-method” and “language” or expects them in the URL.

This XML method then creates a particular XML, based on which the Render Engine can then create the link.

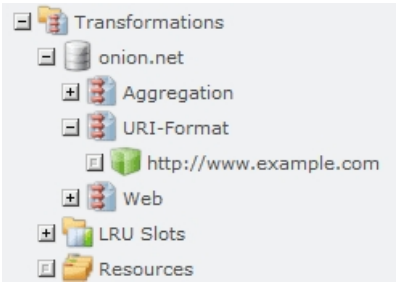
Check whether the attributes are configured accordingly in the web.config of your Render Engine. If this is not the case, add it as described above.

3 Extending transformations

For clarity you should add a new transformationgroup "URI-format". This is usefull to separate the new methods from the ones used for the website-rendering or the aggregation.

If you already have such an transformationgroup and the methods, you could skip this Step.

Add to your tranformationes a new transformationgroup called "URI-format" and under that the datatyps for your Datastructure.
Insert a XML method with the name "uri".



The XML, which the uri()-method has to deliver, looks like that.

```
<path-segments document-name="aktuelle-seite">
  <item>Navigationsebene 1</item>
  <item>Navigationsebene 2</item>
</path-segments>
```

In our example above the attribute "document-name" is the part of the path "dortmund", the current documents name. This is attached behind the ID.

The <item>-elements descripe the parrent pages, like an click path. These elements are optional. In our example they represent the path componentes "ueber-uns/standort". They are inserted between the domain and the parameters respectively the method and the id.

Assume the XML in your uri()-method and adjust it accordingly so that it dynamically fills.

If you are using the Structureabstraction the parentelements match the path parts of the <context> list (without the startpage).

You can maintaine an own element for the path part. The navigation name or the window name are suitable fallback solutions.