TYPO3.Flow - Feature #53080

Support for multiple domains

2013-10-23 17:50 - Bastian Waidelich

Status:	Closed	**Start date:**	2013-10-23
Priority:	Should have	**Due date:**	
Assignee:		**% Done:**	0%
Category:	MVC - Routing	**Estimated time:**	0.00 hour
Target version:			
PHP Version:		**Complexity:**	
Has patch:	No		

Description

Currently the Flow does not support multi-domain setups out of the box:

- There is no concept of "domains" in Flow yet
- The Routing Framework only uses a combination of the (hashed) Request path and method as cache identifier for incoming routes
- How should linking to other domains than the current work? Probably the UriBuilder (and thus all uri/link-ViewHelpers need some "domain" argument)

Approaches

Like done with the filterable request methods (#27117) we could add a new routing configuration:

```
-
  uriPattern: 'some/uri/pattern'
  defaults:
    '@package': 'TYPO3.Flow'
  domain:        'somedomain.tld'
```

But I wonder: Should this support multiple domains? or some regex-/placeholder-syntax?
Besides I'm struggling a bit with adding to many configuration options.. We'll soon need yet another one for the protocol (#44891).

A more generic approach (like I started with https://review.typo3.org/#/c/19742/) would be very flexible. But it has a bad impact on performance because all "filters" can change the caching identifier..

Get basic multi-domain support today

You can get multi-domain Routing working by creating a custom RoutePartHandler that only matches/resolves when the domain is one of the configured. But then you have to disable the Routing Cache in Caches.yaml::

```
Flow_Mvc_Routing_FindMatchResults:
  backend: TYPO3\Flow\Cache\Backend\NullBackend
Flow_Mvc_Routing_Resolve:
  backend: TYPO3\Flow\Cache\Backend\NullBackend
```

NOTE: You should only do so if you **absolutely** know what you're doing. Disabling the Routing Cache can have a really bad impact on the performance.

Another way to get basic support for multiple domains is to use a (sub)context for each domain (Configuration/Production/Domain1/Routes.yaml, Configuration/Production/Domain2/Routes.yaml). But this won't solve the cross-domain-linking issue.

Related issues:

Related to TYPO3.Flow - Bug #54632: route caching should take hostnames into ...	**Resolved**	**2013-12-29**
Related to TYPO3.Flow - Feature #44891: Routes should be able to enforce http...	**New**	**2013-01-28**

History

#1 - 2013-10-24 09:35 - Christopher Hlubek

Well, I think the domain is not as "discrete" as the HTTP method, so users should be able to specify patterns for flexible matching.

What about adding the domain (and protocol?) to the uriPattern and handle that with something like a domain route part?

Example (for any protocol - it's actually a valid absolute URI):

```
-
  uriPattern: '//somedomain.tld/some/uri/pattern'
  defaults:
    '@package': 'TYPO3.Flow'
```

Having dynamic route parts as a part of the host would be awesome:

```
-
  uriPattern: '//{user}.mycloudservice.com/some/uri/pattern'
  defaults:
    '@package': 'TYPO3.Flow'
  routeParts:
    user:
      objectType: 'My\Demo\Domain\Model\User'
```

Wildcards could be implemented with a custom route part with that already (e.g. RegexRoutePartHandler) or we add support for wildcards:

```
-
  uriPattern: '//*.mycloudservice.com/some/uri/pattern'
  defaults:
    '@package': 'TYPO3.Flow'
```

#2 - 2013-10-24 11:41 - Bastian Waidelich

Christopher Hlubek wrote:

Thanks for your feedback, this really helps pushing this forward!

> Well, I think the domain is not as "discrete" as the HTTP method, so users should be able to specify patterns for flexible matching.

Yes, I agree

> What about adding the domain (and protocol?) to the uriPattern and handle that with something like a domain route part?

I once came up with a similar approach, but I can't remember why I dropped the idea back then. I think, this is actually a quite flexible and especially intuitive solution. Although it is not so easy to implement.

> Example (for any protocol - it's actually a valid absolute URI):

I'm not so sure regarding the protocol inside the uriPattern – for this I think an additional option schemes (working like httpMethods) might be more flexible. But on the other side: If we only allow "http://", "https://" and "//" (=both) prefixes we'd have the same flexibility – question is: what is the default protocol when creating URIs?

> Having dynamic route parts as a part of the host would be awesome

Yes, that should work out-of-the-box if we use the regular RoutePart-handling. S.th. like this would also work:

```
-
  uriPattern: '//{@controller}.mydomain.tld'
```

> Wildcards could be implemented with a custom route part with that already (e.g. RegexRoutePartHandler) or we add support for wildcards:

We could use the existing RoutePart-mechanism. Wildcards would be tricky cause you can't define a default. So we can't replace the * when creating URIs. but this would work:

```
uriPattern: '//{subdomain}.mydomain.tld'
defaults:
  'subdomain': 'www'
routeParts:
  'subdomain':
    handler:   'My\Package\RoutePartHandlers\RegexRoutePartHandler'
    options:
      pattern: '/^(www|store|secure)$/'
```

In conjunction with nested subroutes this could be put into one **Routes.Host.yaml** file and be combined like this:

```
uriPattern: '<HostSubRoutes><PathSubRoutes>'
subRoutes:
  'HostSubRoutes':
    package: 'My.Package'
    suffix:  'Host'
  'PathSubRoutes':
    package: 'My.Package'
    suffix:  'Path'
```

in order to reduce duplication.

Some questions remain, though:

- How to deal with the "special" subdomain "www"? www.subdomain.domain.tld is valid. But probably we don't have to react to this differently than to any other (nested) subdomains
- What about installations where Flow/Neos is not running in the web root? http://www.somedomain.tld/flow/this/is/the/path - We should probably skip "/flow" (as we do now), but what about outgoing (absolute) Links?
- Are there other schemes than "http" and "https" that we need to take care of?
- If the scheme is omitted, what should be the behavior for created URIs? (see below for a suggestion)
- How to integrate that into Neos? There the FrontendNodeRoutePartHandler probably needs to be involved as the domain is bound to a node

Examples

Given following route:

```
uriPattern: '//{department}.domain.tld/..'
defaults:
  'department': 'default'
```

This should be the behavior (I think?):

Fluid	Result	Comment
{f:uri.action(action: 'index', absolute: true)}	http://default.domain.tld/..	
{f:uri.action(action: 'index', arguments: '{department: "store"}', absolute: true)}	http://store.domain.tld/..	
{f:uri.action(action: 'index')}	/..	when on **default.domain.tld**, otherwise: http://default.domain.tld/..
{f:uri.action(action: 'index', arguments: '{department: "store"}')}	/..	when currently on **store.domain.tld**, otherwise http://store.domain.tld/..
scheme		
{f:uri.action(action: 'index', absolute: true, scheme: 'https')}	https://default.domain.tld/..	
{f:uri.action(action: 'index', arguments: '{department: "store"}', absolute: true, scheme: 'https')}	https://store.domain.tld/..	
{f:uri.action(action: 'index', scheme: 'https')}	/..	when on https://default.domain.tld, otherwise: https://default.domain.tld/..
{f:uri.action(action: 'index', arguments: '{department: "store"}', scheme: 'https')}	/..	when on https://store.domain.tld, otherwise https://store.domain.tld/..

#3 - 2013-10-25 14:04 - Rafael Kähm

Hello folks,

i have currently no examples for that but is it maybe convinient to split up all domains by contexts(Production/domainname)?

#4 - 2013-10-26 13:44 - Alexander Berl

I basically see two options:
- configure the domain seperately from the uriPattern, which allows to handle paths independant of domain (e.g. Flow in a sub dir of doc root)
- configure the domain with the uriPattern, which allows flexible setup with subroutes as bastian suggests

I think wildcards are not the best thing to do, as you always have to configure the default value to be able to build URIs. Also I think the same behaviour can be achieved with variables anyway

Bastian Waidelich wrote:

> - How to deal with the "special" subdomain "www"? www.subdomain.domain.tld is valid. But probably we don't have to react to this differently than to any other (nested) subdomains

I'd say the latter - don't handle differently

> - What about installations where Flow/Neos is not running in the web root? http://www.somedomain.tld/flow/this/is/the/path - We should probably skip "/flow" (as we do now), but what about outgoing (absolute) Links?

See above, I think it's hard to handle correctly with the uriPattern method and you'd need to add a lot of tricky code to insert the subdir into the path

> - Are there other schemes than "http" and "https" that we need to take care of?

I can't think of any - are there actually any other protocols that are handled with php? Like a rtmp streaming server?

> - If the scheme is omitted, what should be the behavior for created URIs? (see below for a suggestion)

I fully with your suggestions there

> - How to integrate that into Neos? There the FrontendNodeRoutePartHandler probably needs to be involved as the domain is bound to a node

I'm not really into Neos, so I skip that point ;)

#5 - 2014-11-18 10:34 - Bastian Waidelich

- Assignee deleted (Bastian Waidelich)

Sorry, I have to unassign myself. this is currently no blocker for neos, so I have to prioritize other todos. If anyone feels like creating an initial patch, I'm more than happy to help. And in any case I will implement this eventually.
In the meantime a lot of the common scenarios can be covered with HTTP components (
http://docs.typo3.org/flow/TYPO3FlowDocumentation/TheDefinitiveGuide/PartIII/Http.html)

#6 - 2016-05-06 12:53 - Bastian Waidelich

- Status changed from Accepted to Closed

Moved the ticket to https://jira.neos.io/browse/FLOW-453