TYPO3.Flow - Feature # 39740

Status:	Closed	**Priority:**	Should have
Author:	Andreas Wolf	**Category:**	MVC - Routing
Created:	2012-08-12	**Assigned To:**	Bastian Waidelich
Updated:	2013-06-12	**Due date:**	
PHP Version:			
Has patch:	No		
Complexity:			
Subject:	Allow setting fixed values for route parts		

Description

In some situations, it is desirable to have routes that are only applied to certain sets of parameters, e.g. to have some subpackage in a completely different URI namespace:
api/v1/<controller>/<action> -> package MyPackage, subpackage Api<controller>/<action> -> package MyPackage
You could of course use the {@subpackage} key in the route URI here:
{@subpackage}/v1/{@controller}/{@action}
But as soon as you add a second subpackage, this will fail, because both subpackages would be matched by this route.

Setting "api" as the default value of @subpackage would also fail here, as all requests with a controller and an action would be matched by this route then, leading to errors for controllers that do not exist in the Api subpackage.

A solution for this is to allow fixed values for route parts, which are both

1. matched when building the URI (so the route does not match if the $routeValues array passed to resolves() does not contain these fixed values)
2. reconstructed when the URI is matched against the pattern (so they are available when resolving the controller and action)

I propose to add these fixed values to the routeParts array in the configuration, either as a simple value for the route part name or as the value of the "value" subkey:

```
-
name: "option A: fixed value as value of the route part name"
uriPattern: "api/v1/{@controller}/{@action}"
routeParts:
  '@package':    'MyPackage'
  '@subpackage': 'Api'

-

name: "option B: fixed value in the 'value' key"
uriPattern: "api/v1/{@controller}/{@action}"
routeParts:
  '@package':
    value:     'MyPackage'
  '@subpackage':
    value:     'Api'
```

I have implemented option A locally and will provide a patch after ironing out some glitches.

Related issues:

related to TYPO3.Flow - Bug # 43589: Route defaults can be omitted when creat...	**Resolved**	**2012-12-04**

History

#1 - 2012-08-12 22:42 - Andreas Wolf

Gnah, could somebody with the required permissions please fix the formatting?

#2 - 2012-11-27 15:54 - Bastian Waidelich

- Status changed from New to Needs Feedback

- Assigned To set to Bastian Waidelich

Hi Andreas,

sorry for the late reaction. I stumbled upon this issue several times but I never completely got it ;)

What's the difference between your example

```
-
  name: "option A: fixed value as value of the route part name"
  uriPattern: "api/v1/{@controller}/{@action}"
  routeParts:
    '@package':    'MyPackage'
    '@subpackage': 'Api'
```

and

```
-
  name: "option A: fixed value as value of the route part name"
  uriPattern: "api/v1/{@controller}/{@action}"
  defaults:
    '@package':    'MyPackage'
    '@subpackage': 'Api'
```

?

#3 - 2012-11-27 16:06 - Andreas Wolf

The difference is that with defaults, these will be set if the value is not set at all. Therefore, all requests where no package/subpackage is set in the request will go to MyPackage\Api, which is not what I desire. I want it the other way round - only the requests to this subpackage have to match this route, no other requests. For them, I have a different URL scheme.

#4 - 2012-11-27 16:15 - Bastian Waidelich

Andreas Wolf wrote:

> *I want it the other way round - only the requests to this subpackage have to match this route, no other requests.*

Not sure whether I got this now, let me rephrase:
In your example you want the route to match only if the request path is "api/v1/<someController>/<someAction>" **AND** some other request argument (e.g. POST) is array('@package' => 'MyPackage', 'subpackage' => 'Api') ?
Or how would you specify those in your example?

#5 - 2012-11-28 09:45 - Andreas Wolf

Bastian Waidelich wrote:

> *Andreas Wolf wrote:*
>
> > *I want it the other way round - only the requests to this subpackage have to match this route, no other requests.*
>
> *Not sure whether I got this now, let me rephrase:*
> *In your example you want the route to match only if the request path is "api/v1/<someController>/<someAction>" **AND** some other request argument (e.g. POST) is array('@package' => 'MyPackage', 'subpackage' => 'Api') ?*

No, I want all requests to /api/v1/.* to go to the Api subpackage. But I only want links to this subpackage to have a URL /api/v1/... This does not work if I set defaults, as all view helpers etc. where the package and/or subpackage is not set will then have these defaults applied - leading to errors because the controller is not found in the API subpackage.

#6 - 2012-11-28 12:28 - Bastian Waidelich

Andreas Wolf wrote:

> > *In your example you want the route to match only if the request path is "api/v1/<someController>/<someAction>" **AND** some other request argument (e.g. POST) is array('@package' => 'MyPackage', 'subpackage' => 'Api') ?*

> *No, I want all requests to /api/v1/.* to go to the Api subpackage. But I only want links to this subpackage to have a URL /api/v1/...*
> *This does not work if I set defaults, as all view helpers etc. where the package and/or subpackage is not set will then have these defaults applied - leading to errors because the controller is not found in the API subpackage.*

Thanks for your patience, now I'm with you!
I didn't get it at first, because the behavior you describe was meant to be the default.. That is: when creating an URI you have to set all values that are defined in the Routes default (except for @format which is a bit special)..

I don't know when this changed or if it was broken form the start, but right now you can omit default values that are not part of the uriPattern..
Changing Route::compareAndRemoveMatchingDefaultValues() from

```
protected function compareAndRemoveMatchingDefaultValues(array $defaults, array &$routeValues) {
    foreach ($defaults as $key => $defaultValue) {
        if (isset($routeValues[$key])) {
            if (is_array($defaultValue)) {
                if (!is_array($routeValues[$key])) {
                    return FALSE;
                }
                return $this->compareAndRemoveMatchingDefaultValues($defaultValue, $routeValues[$key]);
            } elseif (is_array($routeValues[$key])) {
                return FALSE;
            }
            if (strtolower($routeValues[$key]) !== strtolower($defaultValue)) {
                return FALSE;
            }
            unset($routeValues[$key]);
        }
    }
    return TRUE;
}
```

to

```
protected function compareAndRemoveMatchingDefaultValues(array $defaults, array &$routeValues) {
    foreach ($defaults as $key => $defaultValue) {
        if (!isset($routeValues[$key])) {
            if ($defaultValue === '' || ($key === '@format' && $defaultValue === 'html')) {
                continue;
            }
            return FALSE;
        }
        if (is_array($defaultValue)) {
            if (!is_array($routeValues[$key])) {
                return FALSE;
            }
            return $this->compareAndRemoveMatchingDefaultValues($defaultValue, $routeValues[$key]);
        } elseif (is_array($routeValues[$key])) {
            return FALSE;
        }
        if (strtolower($routeValues[$key]) !== strtolower($defaultValue)) {
            return FALSE;
        }
        unset($routeValues[$key]);
    }
    return TRUE;
}
```

fixes this for me, but it would be quite a breaking change probably..
could you please verify if that works for you?

BTW: To make debugging with fluid easier you can replace line 263 of AbstractViewHelper from throw $exception; to return $exception->getMessage(); – we'll fix this in fluid asap

#7 - 2012-12-18 15:11 - Bastian Waidelich

- Status changed from Needs Feedback to Closed

This should be resolved with #43589 Please reopen/comment if that did not solve your issue

#8 - 2013-04-04 09:41 - Andreas Wolf

- Status changed from Closed to New

I just upgraded our project to recent master and it seems we still have problems.

I've got this route configuration:

```
name: 'Pools'
uriPattern: 'admin/pools(/{@action})'
defaults:
  '@package':    'MyPackageB'
  '@controller': 'Pool'
  '@action':     'index'
```

MyPackageB is a side package for some things that have nothing to do with my main package, so most routes go to the other package.

The problem here is that I have another route configured that matches a path like /admin/pools. Flow assumes that I want the controller "admin" from "MyPackageA", but that's plain wrong, because there is no such controller. Instead, the route given above should match.

Could it be a problem that the route given above is below the route that is used by Flow instead?

#9 - 2013-04-04 10:20 - Bastian Waidelich

- Status changed from New to Needs Feedback

Andreas Wolf wrote:

Hi Andreas,

> *Could it be a problem that the route given above is below the route that is used by Flow instead?*

Yes, sure, "The first route that matches, determines which action will be called with what parameters." – see http://docs.typo3.org/flow/TYPO3FlowDocumentation/TheDefinitiveGuide/PartIII/Routing.html#the-router

Can you please post the route that should not match but does?

#10 - 2013-04-04 10:37 - Bastian Waidelich

BTW: With

```
./flow routing:list
```

you can see in what order the routes are processed. Try including your "MyPackageB" routes before the "MyPackageA" routes or refine the routes to make them more specific

#11 - 2013-05-07 18:41 - Bastian Waidelich

- Status changed from Needs Feedback to Closed

Closing due to lack of feedback (feel free to re-open or comment if this issue is still relevant)

#12 - 2013-05-29 09:45 - Andreas Wolf

In fact, it is almost resolved. I tried it with plain Flow master today (well, mostly, I still have some custom patches applied, but they don't matter for this issue).

All in all #43589 solved most of my problems. One still remains though:

When building a route for these values (taken from System_Development.log):

```
@action => index
@controller => user
@package => mypackage
@subpackage => admin
@format => html
```

with these routes (only the relevant ones)

```
-
 name: 'User admin controller'
 uriPattern: 'admin/user/{user}(/{@action})'
 defaults:
   '@package':    'MyPackage'
   '@subpackage': 'Admin'
   '@controller': 'User'
   '@action':     'index'

-
 name: 'Admin action'
 uriPattern: 'admin/{@controller}(/{@action})'
 defaults:
   '@package':    'MyPackage'
   '@subpackage': 'Admin'
   '@action':     'index'
```

none of the routes matches, but instead a Flow default route is used. When I remove the default routes, exception #1301610453 is thrown.

When invoking admin/user/ manually, it works perfectly - so somehow the router does not seem to match the values to the route, though it should (package and subpackage are fixed as defaults, the controller is given and the action is both given in the route values and in the route's defaults).

#13 - 2013-05-29 11:18 - Bastian Waidelich

- Status changed from Closed to Needs Feedback

Thanks for your feedback.
Just for clarification:
We use the terms

- **match** for the mapping of a URI path to the route values (incoming)
- **resolve** for mapping route values to an URI (outgoing)

In your example the **@package** route value ("qrserver") is not equal to the routes default value "MyPackage".
Was that just a typo in the issue description?
Also your route values specify a **@format** which is not defined in the routes.

#14 - 2013-05-31 11:45 - Bastian Waidelich

- Status changed from Needs Feedback to Closed

Closing again due to lack of feedback. Feel free to comment anyways, I'm monitoring this issue.

#15 - 2013-06-12 17:14 - Andreas Wolf

Bastian Waidelich wrote:

> *Thanks for your feedback.*
> *Just for clarification:*
> *We use the terms - **match** for the mapping of a URI path to the route values (incoming)*
> *- **resolve** for mapping route values to an URI (outgoing)*
>
> *In your example the **@package** route value ("qrserver") is not equal to the routes default value "MyPackage".*
> *Was that just a typo in the issue description?*

I'm sorry, that was in fact a typo.

> *Also your route values specify a **@format** which is not defined in the routes.*

The route values come from Flow internally. I thought that @format would be html by default nevertheless, so I did not really care about this. I added @format: 'html' to the defaults section of all routes now nonetheless, so let's see if it helps.