TYPO3.Flow - Feature # 37604

Status:	Resolved	**Priority:**	Should have
Author:	Karsten Dambekalns	**Category:**	Http
Created:	2012-05-30	**Assigned To:**	Bastian Waidelich
Updated:	2014-12-16	**Due date:**	
PHP Version:			
Has patch:	No		
Complexity:			
Subject:	Make Flow RestController useless by improving MVC stack		

Description

The REST controller provided by FLOW3 should be no longer needed if the MVC stack provides the needed functionality.

Related issues:

related to TYPO3.Flow - Feature # 27117: Bind routes to HTTP request methods	**Resolved**	**2011-05-29**	**2013-04-13**
related to TYPO3.Flow - Feature # 37402: Make request handling RESTfuller	**Resolved**	**2012-05-22**	
related to Base Distribution - Work Package # 45088: Improved REST support	**Resolved**	**2011-05-29**	**2013-04-13**
related to TYPO3.Flow - Feature # 45293: More flexible parsing of body arguments	**Resolved**	**2013-02-08**	**2013-04-13**
related to TYPO3.Flow - Feature # 47137: HTTP method tunneling	**Resolved**	**2013-04-11**	

History

#1 - 2012-05-30 14:04 - Bastian Waidelich

Some ideas/todos:

Routing

- Routes must be "restrictable" to certain request methods (see #27117)
- FLOW3 should provide a set of RESTful Routes in a separate RestRoutes.yaml
  - It must be possible to include sub routes not only by **package** but also by **file path**

Request/Response

- Redirects should keep important request/response headers (e.g. **Accept**)
- Redirects must not output a <meta refresh tag if format !== 'html'
- **Content-Type** response header should be set according to some **$responseContentTypePattern** (e.g. application/@package.@resourceName+@format
- errorAction needs to return the correct status code (400?) for validation errors

#2 - 2012-07-17 11:00 - Marco Falkenberg

Some additional thoughts...

Processing of request content

In case of RESTful webservices the processing of the request's content should be **more advanced and extensible**. For now i.e. XML data is poorly treated by just converting over a SimpleXMLElement to an array. I know that this lack is mentioned by Robert in the comment, but there is no according issue tracker.

The funny thing is we have everything to make this work, because we have the property mapper with its type converters and their customizable configurations. Why don't **allow IANA types AND class types as source types** for a type converter? It's just pulling the responsibility for the property mapper one level up. Then you can implement the handling of XML data with your own XML type converter that i.e. accepts the source types 'application/xml', 'DOMNode', 'DOMElement', 'DOMDocument', 'DOMText'.

I tried it out and it works!

#3 - 2012-07-26 13:53 - Bastian Waidelich

Marco Falkenberg wrote:

> *Some additional thoughts...*

Hi Marco,

very nice ideas. Feel free to share your code ;)

#4 - 2012-12-10 13:39 - Karsten Dambekalns
- Target version changed from 2.0 beta 1 to 2.0

With https://review.typo3.org/11704 we are almost there…

#5 - 2012-12-10 15:26 - Bastian Waidelich
- Target version changed from 2.0 to 2.1

I think, the abstract RestController won't be useless – or at least it's functionality. I like Marcos idea to reuse the type converters somehow. I'll try to get into this for 2.1

#6 - 2013-01-18 15:19 - Marco Falkenberg

Hi Bastian,

another aspect that a REST API should care about is not only to address resources but also their sub-resources. Here are some examples:

```
GET  /products/4711/reviews    (get a list of reviews from the product 4711)
POST /products/4711/reviews    (add a new review to product 4711)
GET  /products/4711/reviews/0815    (get details from review 0815)
PUT  /products/4711/reviews/0815    (to update a review)

Or if reviews are not in the aggregate boundaries defined by a product and have their own repository:
GET  /reviews/0815
PUT  /reviews/0815

To partially update properties of a resource via virtual sub-resources:
PUT /products/4711/description
```

I know this REST-thing is a pretty much complex tasks, because it requires further adjustments to the property mapping and routing. I have made some prototypish experiences to build a REST API. Maybe we could develop some aspects together, if you could manage it. I must say, i don't have any experience with the TYPO3 developing process. I know git, and a bit of gerrit.

#7 - 2013-01-25 17:02 - Bastian Waidelich

Some more missing features and bugs:

Merging of *body* and *GET* arguments does not work

In a PUT request to a resource, the router currently merges request & routing values.
If the request is represented as a identifier string in the URI (e.g. UUID), this does not work, because the router can't merge this:

```
array('product' => '<uuid>');
```

```
array('product' => array('title' => 'changed title'))
```

A solution might be to move the parsing of *argument body* from the request to a later place (in the action controller right before property mapping takes place?)

Parsing of body arguments is not very flexible

Parsing of body arguments (e.g. xml, json) currently takes place in Http\Request and is hard-coded there.
Besides from being a bit buggy (e.g. you currently have to specify two root nodes like <root><product><title>changed title</title></product></root> where a common REST request would expected to be just <title>changed title</title>) the behavior can't be influenced easily because it happens so early in the request handling

media type negotiation does not work for custom types

E.g. an Accept header of "application/hal+json" is not correctly resolved to */json

How to modify collections

Imagine a Resource "product" that has n "tags".
How do you remove a tag from the collection w/o deleting the actual tag resource?

You could of course update the product and transfer the complete tag collection w/o the tag to remove:

```
PUT products/1 {tags: [ {"__identity": "2"}, {"__identity": "3"}]}
```

but that's not very practical for large collections.

```
DELETE tags/4
```

would remove the tag globally, but would this make sense:

```
DELETE products/1/tags/4
```

Maybe this can be resolved with different HTTP methods (such as PATCH or DELETE) but they're currently not supported by Flow

#8 - 2013-02-01 19:05 - Bastian Waidelich

- Subject changed from Make FLOW3 RestController useless by improving MVC stack to Make Flow RestController useless by improving MVC stack

#9 - 2013-05-21 13:28 - Robert Lemke

- Target version deleted (2.1)

#10 - 2013-05-28 18:43 - Bastian Waidelich

FYI I just pushed a first WIP for improved request argument handling: https://review.typo3.org/#/c/21134/

#11 - 2014-12-16 11:50 - Bastian Waidelich

- Status changed from Accepted to Resolved

- % Done changed from 0 to 100

~~With version 2.3 (and partly 2.2) The RestController is deprecated and no longer required~~ The RestController is still not deprecated because we're missing a nicely integrated and fully documented alternative yet. But in general it should not be needed because everything can be achieved with a couple of routes.