TYPO3.Flow - Feature # 47551

Status:	Rejected	**Priority:**	Should have
Author:	Alexander Berl	**Category:**	Http
Created:	2013-04-24	**Assigned To:**	
Updated:	2013-05-21	**Due date:**	
PHP Version:			
Has patch:	No		
Complexity:			
Subject:	Allow usage of links for destructive actions again		

Description

With the new changes that prevent unsafe methods to be destructive (no persistence), it is currently impossible to create action links, that execute unsafe methods.

A use case would be a backend of some sort, where you have a list of "products" (or other entities), each with a set of icons linking to methods such as "copy", "rename", "delete", etc.
As it is, one would have to create a form for each such action, having to deal with lots of html overhead and bad stylability or use javascript/XHR to work around that. In any way, it's a cumbersome solution to such a common requirement.

I therefore suggest, to:

- allow GET methods to tunnel other request methods
- extend the link.action viewhelper with a method parameter, which allows to override the request method through the request method tunneling feature
- add csrf token to action links automatically, if the overridden method is unsafe

This would allow unsafe action links to be generated explicitly (thus conciously) but still without too much boilerplate work.

Please give feedback.

Related issues:

related to TYPO3.Flow - Feature # 47137: HTTP method tunneling	**Resolved**	**2013-04-11**

History

#1 - 2013-04-24 16:02 - Bastian Waidelich

Even though it is done a lot (we have caught ourselves here) it is really bad practice to have "delete" links in an application.
Per definition crawlers (including search engines) are allowed to follow links and the only reason that this doesn't end up in loss of data, is because those delete links are usually behind a login.. but imagine a delete link on a wiki.

Following the notion of "safe requests" will us allow to do great optimizations in the future such as:

- Running persistence in "readonly" mode for GET
- Partly disable validation for GET
- No CSRF hassle
- **Smarter caching** of GET requests

Don't get me wrong. Flow is known to be "opinionated", but this doesn't mean you're forced to do it the "flow" way. But I don't agree with your approaches for following reasons:

> *allow GET methods to tunnel other request methods*

This would circumvent the whole "safe method" approach.

> *extend the link.action viewhelper with a method parameter, which allows to override the request method through the request method tunneling feature*

That wouldn't work out for the above reason.
But what we should provide is documentation on how to adjust your application:

The preferred way is probably to replace links with little forms where possible:
from:

```
1<f:link.action action="delete">delete</f:link.action>
```

to:

```
1<f:form action="delete"><f:form.button>delete</f:form.button></f:link.action>
```

If you really need links (I'd be interested in use cases) you could set some (data) attribute on the link and unobtrusively turn them into POSTs on click via JavaScript:

In Ruby on Rails the **link_to** helper has an argument **:method** that we could adopt to make this a bit easier.
So this:

```
1<f:link.action action="delete" method="delete">Remove</f:link.action>
```

would result in:

```
1<a href="foo/bar" data-method="delete" rel="nofollow">Remove</a>
```

The JavaScript RoR uses to turn this into a form on the fly is quite interesting, too:
https://github.com/rails/jquery-ujs/blob/master/src/rails.js#L151

And last but not least (or maybe also least):

You can keep everything as is and change your action from

```
1/**
2 * @param Foo $foo
3 **/
4public function deleteAction(Foo $foo) {
```

```
5  $this->fooRepository->delete($foo);
6}
```

to:

```
1/**
2 * @param Foo $foo
3 * @Flow\SkipCsrfProtection
4 **/
5public function deleteAction(Foo $foo) {
6  $this->fooRepository->delete($foo);
7  $this->persistenceManager->persistAll();
8}
```

Be warned though, that this circumvents CSRF protection (which might not be an issue here).

BTW: There is a brand new CsrfToken ViewHelper in Fluid to keep CSRF protection working for non-fluid forms & javascript POST requests

#2 - 2013-04-24 17:34 - Alexander Berl

That's pretty much exactly what I thought of. A JavaScript helper would be cool too, just not sure how that could be properly bundled with flow, so that it works out of the box (automatic JS injection when a link.action method parameter is given?).

#3 - 2013-04-25 10:37 - Bastian Waidelich

doh, I just realized that I replaced my long comment from yesterday by this addition.. does anyone has the original text still (maybe in your email inbox)? would be nice to add this again for others with the same question..

#4 - 2013-04-25 11:20 - Bastian Waidelich

Bastian Waidelich wrote:

> *doh, I just realized that I replaced my long comment from yesterday by this addition.. does anyone has the original text still (maybe in your email inbox)? would be nice to add this again for others with the same question..*

Thanks to Marc I could add it to the above comment again. History rewriting FTW! ;)

#5 - 2013-05-21 12:19 - Robert Lemke

- Status changed from New to Rejected

- Target version set to 2.0

Closing this issue. Feel free to re-open if you end up with a different conclusion.