TYPO3.Flow - Feature # 34133

Status:	New	**Priority:**	Could have
Author:	Jacob Floyd	**Category:**	Property
Created:	2012-02-21	**Assigned To:**	
Updated:	2012-02-21	**Due date:**	
PHP Version:			
Has patch:	No		
Complexity:			
Subject:	RFC: Handle Semicolons in Path part of URIs as Scoped Path Parameters		

Description

Scoped Path Parameters

The URI RFCs hint at what I think will make an awesome, unique, feature for Flow3, and Typo3 Phoenix: Scoped Path Parameters. It's a powerful feature that could allow all sorts of search faceting, matrices, or scoped parameters available for use in packages.

One of the possible uses for a path parameter in Typo3 might be to view historical versions of pages or branches of the page tree. You could set the version parameter on a particular part of the hierarchical path (not necessarily the final segment that identifies the page) and get past published versions of a page in that branch. Workspaces are great for queuing future work and pushing it through the required workflow stages, but workspaces are awkward for managing past versions. But I digress - what exactly is a path parameter?

What is a *Path Parameter*?

> *A URI path pa-ram-e-ter is part of a path seg-ment that oc-curs after its name. Path pa-ram-e-ters offer a unique op-por-tu-ni-ty to con-trol the rep-re-sen-ta-tions of re-sources. Since they can't be ma-nip-u-lat-ed by stan-dard Web forms, they have to be con-struct-ed out of band. Since they're part of the path, they're se-quen-tial, un-like query strings. Most im-por-tant-ly, how-ev-er, their be-haviour is not ex-plic-it-ly de-fined. - HTTP Path Parameter Syntax*

Consider the latest version of the URI spec (3986) which says:

> *URI producing applications*
> *often use the reserved characters allowed in a segment to delimit*
> *scheme-specific or dereference-handler-specific subcomponents. For*
> *example, the semicolon (";") and equals ("=") reserved characters are*
> *often used to delimit parameters and parameter values applicable to*
> *that segment. The comma (",") reserved character is often used for*
> *similar purposes. For example, one URI producer might use a segment*
> *such as "name;v=1.1" to indicate a reference to version 1.1 of*
> *"name", whereas another might use a segment such as "name,1.1" to*
> *indicate the same. Parameter types may be defined by scheme-specific*
> *semantics, but in most cases the syntax of a parameter is specific to*
> *the implementation of the URI's dereferencing algorithm.*
> - *3986*

Though 3986 has an EBNF definition that allows for parameters in the path (like "name;v=1.1"), I think it's clearer to see in the previous version (2396) where it shows that each path_segment (separated by a "/" can be followed by parameters.
[...]

Examples

So, In a path like /foo/bar/baz, you could add parameters that apply to a particular branch of the page tree:
/foo;v=1.1/bar;authgroup=employee/baz;views=map,calendar

This means that http://www.blah.com/some;param1=foo/crazy;param2=bar/path.html is a perfectly valid url (see this article and this page on the subject).

Other examples

- /products;filter=family-friendly/videos;rating=pg?q=dragon

- /foo/abc/bar/def
  /foo;v=2.8/abc/bar/def;diff=with-current
  /foo;L=en/abc/bar;L=de/def
  /foo/abc;w=draft/bar/def;diff=with-other-authors

- /map;type=topo/south-america/Brasil/São-Pualo;type=subway,cycle

- /users;auth-group=super/John-Smith
  is obviously a different view of the same resource:
  /users/John-Smith
  only this one is the public view, and the other is the super-user view

If I were to implement amazon.com's faceted search with path parameters, it might do this:
With Path Params:
http://www.amazon.com/Movies-TV;format=DVD,Blu-Ray;genre=drama;stars=3+;price=$0.01-$20/Movies;actor=Christopher-Lee;dir
ctor=Alan-Gibson

> *Original:*
> *http://www.amazon.com/gp/search/ref=sr_nr_p_36_0?bbn=2649512011&qid=1329629982&rh=n%3A2625373011%2Cp_n_form*
> *t_browse-bin%3A2650304011%7C2650305011%2Cn%3A%212625374011%2Cn%3A2649512011%2Cp_72%3A3014476011%2Cp_n_th*
> *Cp_n_theme_browse-bin%3A2650368011%2Cp_36%3A3052254011&rnid=3052254011&low-price=0.01&high-price=20&x=0&y=0#/ref=s*
> *0#/ref=sr_nr_p_n_feature_three_br_2?rh=n%3A2625373011%2Cp_n_format_browse-bin%3A2650304011%7C2650305011%2Cn%3A%2*
> *%3A%212625374011%2Cn%3A2649512011%2Cp_72%3A3014476011%2Cp_n_theme_browse-bin%3A2650368011%2Cp_36%3A1-20*
> *3A1-2000%2Cp_n_feature_three_browse-bin%3A2651257011&bbn=2649512011&ie=UTF8&qid=1329630193&rnid=2651254011*

Implementation

- TYPO3\FLOW3\Property\DataType\Uri breaks things down into the different URI components
- TYPO3\FLOW3\MVC\Web\Routing\UriBuilder deals with URIs throughout Flow3

UriBuilder\setArguments allows prefixed query args: array('prefix1' => array('foo' => 'bar')) gets "&prefix1[foo]=bar". This could also be done with a path parameter (which intrinsically has a 'scope', ie, earlier pages or templates in the rootline won't be able to use a parameter set on their children. The parameter applies only from that level on.) &prefix1[foo]=bar might become my-controller/prefix1;foo=bar/fun-stuff.

Routing

I think that Flow3 needs to handle the path parameters by default. Packages should not be required to create special routes just to handle these parameters - they should be loaded in some kind of array that is accessible from within the package. The default handling could be changed if required for a particular installation (see defaults below).
So, I'd imagine that routing would happen like this (or something like it):
1. Separate a given URI (possibly an requested resource) into the various parts, like now: host, user, path, query ...
2. Additionally parse the path segment to place all of the path params in a path_param array - something that helps respect the 'scoped' nature of these parameters - They are much more specific than the general query string parameters.
3. Strip all of the path parameters from the path and process routing as usual.
   - This is important - routing matches only on the path-segments, not on the path parameters. A new kind of routing rule could be used to redirect a request based on one or more of the path parameters, but the requested resource **does not change** based on the params. The view might look different, or have additional features, but the path sans parameters and path with parameters identifies *the **same** node*.

Defaults and Security

Accessing the same node, no matter the path parameters is important. Be default, or at least to begin with, path parameters should be detected, stripped, and the request should be redirected with a '302 Found' to the same URL sans the parameters. Then when someone begins to use path parameters in their package, the package can enable them - probably with a special routing rule that turns them on, but only for a particular branch or segment of the path and/or tree.

Back to the point of path parameters not changing the node in the content tree: Allowing redirection to other nodes could introduce some security issues with url spoofing that would not be fun to deal with. Right now, the most common use of the semicolon is to hack insecure software. Tomcat uses the path parameter to pass JSESSIONID when cookies can't easily be included in some request. This Tomcat feature has been the basis for hacking and phishing as some browsers didn't understand the semicolon. To make sure Flow3 doesn't have the same problem - I think support for path parameters is a must, even if all that's done is drop them.

Final thoughts

Even though some browsers might have issues (the articles I saw online were somewhat dated, so the issues may have been resolved) I think Flow3 should support this, because browsers are not the only content consumers around. Flow3 can be the basis for some robust APIs that are merely for inter-server communication. Browsers also don't support HTTPs PUT or DELETE on form elements, but Flow3 is getting amazing support for those... Please Support Path Parameters.

Another important path parameters side effect (hopefully) would be improved and more robust caching of pages that require authentication (say of a particular usergroup).

As a side note, the specs also indicate that semicolons can be used in place of ampersands in the query string. I don't know if Flow3 supports that right now, but I think it should. (If needed, I can file another feature request about that.)

History

#1 - 2012-02-21 11:36 - Karsten Dambekalns

Without having read everything yet, I'd like to point to the arg_separator.* directives of PHP (see http://www.sitepoint.com/php-and-standards-arg_separatoroutput/ for a nice article outlining some of the confusion around those). Whatever we do, **if** the server has a setting to use □ as the output separator, we should follow suit, IMHO.

How would that fit together?

#2 - 2012-02-21 11:52 - Jacob Floyd

…moved to original description.

#3 - 2012-02-21 12:06 - Jacob Floyd

Karsten Dambekalns wrote:

> *Without having read everything yet, I'd like to point to the arg_separator.* directives of PHP (see http://www.sitepoint.com/php-and-standards-arg_separatoroutput/ for a nice article outlining some of the confusion around those). Whatever we do, **if** the server has a setting to use □ as the output separator, we should follow suit, IMHO.*
>
> *How would that fit together?*

Do we really want to support using session.use_trans_sid? From that article, and others I've read, it sounds like it'll give invalid urls within HTML docs (using & instead of &amp;. Overall, session.use_trans_sid sounds like it'll cause us a lot more grief than its worth. I'd rather Flow3 take care of sticking a session ID in the url (as a path param, of course) if that's necessary.

I also think that using arg_separator.output would be a very clean way to have an installation use ';' instead of '&'. Flow3 would just use that when generating or working with the query string. Even though session.use_trans_sid is disabled, Flow3 should still be able to grab this setting, or ignore it by choosing it's own way.

However, I think arg_separator.output is only for the query string. Any Path Parameter handling we do will end up being part of flow3, as not many (none that I know of) apps support these scoped parameters.

#4 - 2012-02-21 15:17 - Karsten Dambekalns

Hi.

Jacob Floyd wrote:

> *Do we really want to support using session.use_trans_sid?*

I don't give a dime about trans_sid :)

> *I also think that using arg_separator.output would be a very clean way to have an installation use ';' instead of '&'. Flow3 would just use that when generating or working with the query string.*

Exactly.

Interesting would be what happens if arg_separator.input is set to ';' - FLOW3 should still work, of course ;)