TYPO3.Flow - Bug #56573

Major Feature # 56602 (New): Handling Of Multi Identity Entities

Converting by Flow\Identity

2014-03-05 14:08 - Carsten Bleicker

Status:	New	**Start date:**	2014-03-05
Priority:	Should have	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Persistence	**Estimated time:**	0.00 hour
Target version:			
PHP Version:		**Complexity:**	
Has patch:	No		

Description

i am receiving records from a third party by json rest api.
my entities getting a property $foreignIdentity wich is the id of third party.
i would expect that the converter finds these entities and converts them by any given identity.
maybe throw exception if its not unique. but it could be prevented by a unique constraint.

i would expect that 2.) works:

1.) http://acme.local/foo/fc700953-8911-9432-542e-968b7bff43e3
Status 200 and flow persistence identifier is converted to concrete object

2.) http://acme.local/foo/5317133a3e1bb <-- foreignIdentity wich is also a @Flow\Identity
Status 500
#1297933823: Object with identity "5317133a3e1bb" not found.

class looks like this:

// @Flow\Entity
class Foo{

/**

- @var string
- @Flow\Identity
  */
  protected $foreignIdentity;

}

my routing:
-
name: 'Read Action by property "foreignIdentity" or "flow_persistence_identifier" as {foo}'
uriPattern: 'foo/{foo}'
httpMethods: ['GET']
defaults:
'@package': 'Acme.Foo'
'@controller': 'Foo'
'@action': 'show'
'@format': 'json'

i dont want to bind the rest api to the flow persistence identifier because third party users should use their own identity in combination with their workspace as unique constraint over the fields foreignIdentity_workspace.

History

#1 - 2014-03-05 15:00 - Bastian Waidelich

- Category set to Persistence

- Status changed from New to Needs Feedback

- Assignee set to Bastian Waidelich

Hi Carsten,

please mind the difference between **identity** and **identifier**!
The following should work (untested):

```
use Doctrine\ORM\Mapping as ORM;

/**
 * @Flow\Entity
 */
class Foo {

    /**
     * @var string
     * @ORM\Id
     */
    protected $foreignIdentity;

}
```

Please report back so we can close/fix the issue

#2 - 2014-03-05 15:17 - Carsten Bleicker

Bastian Waidelich wrote:

> Hi Carsten,
>
> please mind the difference between **identity** and **identifier**!
> The following should work (untested):
> [...]
>
> Please report back so we can close/fix the issue

Doesnt @ORM\Id results in a unique identity?
But my domain needs: @ORM\Table(uniqueConstraints={@ORM\UniqueConstraint(name="workspace_id",columns={"workspace", "id"})})
And as far i can see there is already an converting of multimple @Flow\Identity usage and converting them.
But seems only to work with array containing identities but not with identity as string?
@see \TYPO3\Flow\Property\TypeConverter\PersistentObjectConverter::findObjectByIdentityProperties

Thanks for help!

#3 - 2014-03-05 15:27 - Alexander Berl

(copy of my maillist answer):

I had to take some deep looks in the Flow code to understand, but I think I know what the problem is.

First of all, there is a difference between the persistence identifier (Persistence_Object_Identifier magic attribute or class attribute annotated with @ORM\Id) and an @Flow\Identity annotated attribute.
The latter is not connected to the database primary key, but is supposedly used to identify entities by alternative unique attributes.

Now what happens in 2.) is this:

Since you don't use the identity route part handler in your route configuration, the external identity string is just forwarded directly to the property mapping/type converter.

However, the PersistentObjectTypeConverter expects a string type identifier to be the persistence identifier and tries to retrieve the object by that.
This will obviously not match, as the primary key is the Persistence_Object_Identifer.

The PersistentObjectTypeConvert only falls back to retrieving the object by the additional identity when the value is an array.
This may be arguable behaviour, but so far is supposed to work if you use the IdentityRoutePart configuration, to tell the Routing that you expect an entity that is given by its identifier, which would in turn be converted to array('__identity' => $value), so the TypeConverter would work as expected.

Now, there is another problem with the IdentityRoutePartHandler::getObjectIdentifierFromPathSegment:
If you don't set an Uri pattern, it in turn will also try to check the given identity against the persistence identity (which again fails).
If you do set an Uri pattern (or have your entity annotated with @Flow\Identity attributes), it only works if the route part is cached, e.g. you created an URI to that entity beforehand.

I guess that's something that has to be discussed with the core developers, as the behaviour is not quite consequent, but too deep to give a quick solution.

#4 - 2014-03-05 15:33 - Alexander Berl

@Bastian I think this distinction is not well documented, but apart from that, the behaviour is also inconsequent as explained above. I think Carstens use case, though rare, is valid and it should be possible to use a single **identifier** (as to my understanding it would also work if he used a combined **identifier**) to route an entity, while it still has a different **identity** (e.g. magic persistence identifier).

#5 - 2014-03-05 15:38 - Carsten Bleicker

This is so complicated :/
I have currently no idea how there could be a rest api wich acts for different "warenwirtschaftssysteme" (sorry, dont know the word in english) using the same id "123" wich should be converted automaticaly. access is done by query rewrite f.e. by entity acl:
OutsideOfWorkspace: 'this.workspace != current.restApiContext.workspace'

#6 - 2014-03-05 17:49 - Bastian Waidelich

I agree that the distinction between identity and identifiers could made clearer (feel free to open an issue in the documentation tracker). But I don't think the behavior is inconsequent:
Flow adds the magic **Persistence_Object_Identifer** only if you don't specify an identifier yourself (using the **ORM\Id** annotation).
If the property mapper would look at identifier **and** identity properties to convert an object, the result would not be deterministic.

@Carsten if you don't want to make the **foreignIdentity** the **Id** of your domain model for some reason, what you're looking for is the so called "object route parts" (see http://docs.typo3.org/flow/TYPO3FlowDocumentation/stable/TheDefinitiveGuide/PartIII/Routing.html#object-route-parts).
Try

```
-
  name: 'Read Action by property "foreignIdentity" or "flow_persistence_identifier" as {foo}'
  uriPattern: 'foo/{foo}'
  defaults:
    '@package': 'Acme.Foo'
    '@controller': 'Foo'
    '@action': 'show'
    '@format': 'json'
  routeParts:
    'foo':
      objectType: 'Your\Package\Domain\Model\Foo'
      uriPattern: '{foreignIdentity}'
  httpMethods: ['GET']
```

You can have this route **in addition** to your existing one. But I wouldn't recommend to use "foo/<UUID>" and "foo/<Foreign_Identifier>" side by side as it contradicts the addressability constraint of REST

#7 - 2014-03-05 17:50 - Bastian Waidelich

BTW: Given the difference between "identifier" and "identity" the property "foreignIdentity" should probably be called "foreignIdentifier"

#8 - 2014-03-06 12:44 - Alexander Berl

Bastian Waidelich wrote:

> I agree that the distinction between identity and identifiers could made clearer (feel free to open an issue in the documentation tracker). But I don't think the behavior is inconsequent:
> Flow adds the magic **Persistence_Object_Identifer** only if you don't specify an identifier yourself (using the **ORM\Id** annotation).

So far that's all correct and valid.

> If the property mapper would look at identifier **and** identity properties to convert an object, the result would not be deterministic.

But that's kind of what it already does, if I'm not mistaken:

```
    protected function fetchObjectFromPersistence($identity, $targetType) {
        if (is_string($identity)) {
            $object = $this->persistenceManager->getObjectByIdentifier($identity, $targetType);
        } elseif (is_array($identity)) {
            $object = $this->findObjectByIdentityProperties($identity, $targetType);
        }
        ...
```

if the given **identity** is a simple string, it uses the persistence **identifier** (ie. the magic one or the field(s?) annotated as **@ORM\Id**). If however, the

identity is an array, it uses the "identity properties" of the object, i.e. the fields that are annotated as **@Flow\Identity**.

> @Carsten if you don't want to make the **foreignIdentity** the **Id** of your domain model for some reason, what you're looking for is the so called "object route parts" (see http://docs.typo3.org/flow/TYPO3FlowDocumentation/stable/TheDefinitiveGuide/PartIII/Routing.html#object-route-parts).
> Try
> [...]

If I follow the code correctly, the uriPattern is not needed to be set, as it automatically falls back to the @Flow\Identity properties if it is NULL. In either case, the IdentityRoutePartHandler will only use the objectPathMappingRepository to find the actual identifier of the object to use.
This is latest where the real problem sits IMO: For a REST-API it will not work, unless you also create URIs for all entities beforehand.

#9 - 2014-03-06 13:26 - Christian Müller

- Parent task set to #56602

#10 - 2014-03-06 14:36 - Bastian Waidelich

Alexander Berl wrote:

Thanks for the update.

> if the given **identity** is a simple string, it uses the persistence **identifier** (ie. the magic one or the field(s?) annotated as **@ORM\Id**). If however, the **identity** is an array, it uses the "identity properties" of the object, i.e. the fields that are annotated as **@Flow\Identity**.

You're right - that's misleading if not wrong. We should look into it.

> If I follow the code correctly, the uriPattern is not needed to be set, as it automatically falls back to the @Flow\Identity properties if it is NULL.

True, that has only been fixed recently

> In either case, the IdentityRoutePartHandler will only use the objectPathMappingRepository to find the actual identifier of the object to use.

Yes - this is still a bug / missing feature of the IdentityRoutePart. I created an issue for that and take care of it asap: #56608
But...

> This is latest where the real problem sits IMO: For a REST-API it will not work, unless you also create URIs for all entities beforehand.

Allow me to split hairs: A fully RESTflui client must never construct URIs by itself but the server should, so this should not be a problem. But that's no reason to fix the bug mentioned above of course.

Anyways, using the "object routing" is just one approach. You could also just create a **FooRoutePartHandler** that can convert an incoming "foreignIdentifier" to an instance of foo vice versa.
It's very easy:

```
<?php
namespace Your\Package\Routing;

use TYPO3\Flow\Annotations as Flow;
use TYPO3\Flow\Mvc\Exception\InvalidRoutePartValueException;
use TYPO3\Flow\Mvc\Routing\DynamicRoutePart;
use Your\Package\Domain\Model\Foo;
use Your\Package\Domain\Repository\FooRepository;

class FooRoutePartHandler extends DynamicRoutePart {

    /**
     * @Flow\Inject
     * @var FooRepository
     */
    protected $fooRepository;

    /**
     * @param string $value
     * @return boolean
     */
    protected function matchValue($value) {
        $foo = $this->fooRepository->findOneByForeignIdentifier($value);
        if ($foo === NULL) {
            return FALSE;
        }
        $this->value = array('__identity' => $this->persistenceManager->getIdentifierByObject($foo));
        return TRUE;
    }

    /**
     * @param mixed $value
     * @return boolean
     */
    protected function resolveValue($value) {
        if (!$value instanceof Foo) {
            return FALSE;
        }
        $this->value = $value->getForeignIdentifier();
        return TRUE;
    }
}
```

The corresponding route would look like:

```
-
  name: 'Read Action by property "foreignIdentity" or "flow_persistence_identifier" as {foo}'
  uriPattern: 'foo/{foo}'
  defaults:
    '@package': 'Acme.Foo'
    '@controller': 'Foo'
    '@action': 'show'
    '@format': 'json'
  routeParts:
    'foo':
      handler: 'Your\Package\Routing\FooRoutePartHandler'
  httpMethods: ['GET']
```

(untested)

#11 - 2014-03-06 15:20 - Alexander Berl

Thanks for creating the issue for the objectPathMapping!

Bastian Waidelich wrote:

> > This is latest where the real problem sits IMO: For a REST-API it will not work, unless you also create URIs for all entities beforehand.
>
> Allow me to split hairs: A fully RESTflui client must never construct URIs by itself but the server should, so this should not be a problem. But that's no reason to fix the bug mentioned above of course.

Ah yes, very good point indeed, though probably more theoretical than practical.

#12 - 2014-03-09 12:59 - Carsten Bleicker

I dont understand how this routing matching stuff works.
Is that Rocket Science?
@Bastian:
resolveValue is used to build up the value in link generation?
matchValue is used for the opposite (uri comes in?)

i need this:
incoming uri: rest/show/acme.foo/123

i need to get the __identity by acme.foo/123.
so in think this is a routePart, right?

you shows an example but thats only a single value, in yours the "foreignIdentity".
i need to route a combination of workspace/foreignIdentity.

why isnt this example from documentation working for me?
-
name: 'Single post actions'
uriPattern: 'posts/{post}/{@action}'
defaults:

'@controller': 'Post'
routeParts:
post:
objectType: 'TYPO3\Blog\Domain\Model\Post'
uriPattern: '{date:Y}/{date:m}/{date:d}/{title}'

in my case:
-
name: 'Single post actions'
uriPattern: '{@action}/{artist}'
defaults:
'@package':   'Foo.Bar'
'@controller': 'Artist'
'@format':   'json'
routeParts:
artist:
objectType: 'Foo\Bar\Domain\Model\Artist'
uriPattern: '{workspace}/{foreignIdentity}'

#13 - 2014-04-09 13:10 - Bastian Waidelich

- Status changed from Needs Feedback to New

- Assignee deleted (Bastian Waidelich)

- Priority changed from Must have to Should have

Carsten Bleicker wrote:

> I dont understand how this routing matching stuff works.
> Is that Rocket Science?

I agree that it's not easy to grasp because of the inherent complexity of routing. But it's not rocket science once you got the hang of it.

> resolveValue is used to build up the value in link generation?
> matchValue is used for the opposite (uri comes in?)

Exactly, maybe the doc comments on the RoutePartInterface[1] also help a bit.

> i need this:
> incoming uri: rest/show/acme.foo/123
>
> i need to get the __identity by acme.foo/123.
> so in think this is a routePart, right?

"acme.foo/" is part of the __identity?
When the route is created it splits up the **uriPattern** into **static** and **dynamic** route parts.
The uriPattern "posts/{post}/{@action}" consists of 3 route parts:

- "posts/" a static route part that only matches the request path substring "posts/"
- "{post}" a dynamic route part that (by default) matches everything until a **splitString** which is a slash ("/") in this case
- "{@action}" the same

> why isnt this example from documentation working for me?
> -
> name: 'Single post actions'
> uriPattern: 'posts/{post}/{@action}'
> defaults:
> '@controller': 'Post'
> routeParts:
> post:
> objectType: 'TYPO3\Blog\Domain\Model\Post'
> uriPattern: '{date:Y}/{date:m}/{date:d}/{title}'
>
> in my case:
> -
> name: 'Single post actions'
> uriPattern: '{@action}/{artist}'
> defaults:
> '@package':   'Foo.Bar'
> '@controller': 'Artist'

```
'@format':     'json'
routeParts:
artist:
objectType: 'Foo\Bar\Domain\Model\Artist'
uriPattern: '{workspace}/{foreignIdentity}'
```

Not sure, it looks right, but you should provide more context. But maybe the mailing list is a better place to discuss this.

I unassign myself for now because I can't judge the actual issue, maybe Karsten knows more?

[1] https://git.typo3.org/Packages/TYPO3.Flow.git/blob/HEAD:/Classes/TYPO3/Flow/Mvc/Routing/RoutePartInterface.php