TYPO3.Flow - Feature # 26943

Status:	Needs Feedback	**Priority:**	Should have
Author:	Kevin Ulrich Moschallski	**Category:**	I18n
Created:	2011-05-20	**Assigned To:**	Karsten Dambekalns
Updated:	2013-05-21	**Due date:**	
PHP Version:			
Has patch:	No		
Complexity:			
Subject:	Add i18n support to domain models		

Description

Hi,

i found a doctrine Behavioral Extension which support annotation bases translation for domain models. Look at it here:

http://www.doctrine-project.org/blog/doctrine2-behavioral-extensions

It also has another interesting extension for updating a timestamp property every time an property is changed in a model. Don't know if FLOW3 supports something already, just for notice.

I tried to implement this extension myself in the EntityManagerFactory, but without luck. But i think i18n support for domain models is a major feature missing for beta1.

Regards,
Kevin

History

#1 - 2011-08-04 08:50 - Sebastian Kurfuerst

- Tracker changed from Bug to Feature

#2 - 2011-11-22 00:22 - Karsten Dambekalns

- Category set to I18n
- Priority changed from Must have to Should have
- Target version set to 1.1
- Has patch set to No

#3 - 2012-05-07 22:31 - Karsten Dambekalns

- Status changed from New to Needs Feedback
- Assigned To set to Karsten Dambekalns

The major question to answer before this can be implemented is how it should be used from a developers point of view. How would you like to handle i18n in your models?

#4 - 2012-05-09 22:19 - Alexander Berl

What I would need, is for one or more translations of an object property to be set with a web form.
The use case would be for example a shop system, where the owner can create products in multiple languages without having to switch language between updates.

As a developer I would then like to be able to just create a form that looks somewhere like this:

```
<f:form action="create" name="product" object="{product}">
  English name: <f:form.textfield property="name" locale="en" />
  German name: <f:form.textfield property="name" locale="de" />
  ...
</f:form>
```

Alternatively, the locale could be part of the property path maybe, though that would not be the cleanest solution. On the other hand it also provides an easy option of *displaying* multiple translations of one property in a single template without having to use viewhelpers.

```
<div>English name: {product.en.name}</div>
<div>German name: {product.de.name}</div>
```

As I understand the behavorial extensions, it would not be able to set multiple translations without persisting the entities inbetween, e.g.:

```
$article->setTranslatableLocale('de_de'); // change locale
$article->setTitle('my title in de');
$article->setContent('my content in de');

$article->setTranslatableLocale('en_US'); // change locale
$article->setTitle('my title in en');
$article->setContent('my content in en');
```

This would lead to only the english translation being persisted, unless an explicit persist call is done between. A solution could possibly be to have a translation aspect, that changes setters/getters to write to/read from a hashmap of locale->value and having setTranslatableLocale change the current hashmap index.

Other than that, being able to to just use normal getters/setters of an entity together with an setLocale call seems like a perfectly transparent way of handling translations in Code.

I also like the timestampable extension, though the same can be achieved with database triggers, it makes sense to have such behaviour abstracted away from the db in a good DDD manner.

#5 - 2012-05-21 16:18 - Karsten Dambekalns

- Target version changed from 1.1 to 2.0 beta 1

#6 - 2012-12-10 13:31 - Karsten Dambekalns

- Target version changed from 2.0 beta 1 to 2.1

#7 - 2013-05-21 13:28 - Robert Lemke

- Target version deleted (2.1)