TYPO3.Flow - Feature # 39788

Status:	New	**Priority:**	Could have
Author:	Alexander Berl	**Category:**	Validation
Created:	2012-08-14	**Assigned To:**	
Updated:	2012-08-14	**Due date:**	
PHP Version:			
Has patch:	No		
Complexity:			
Subject:	RFC: Repository based NotExistsValidator		

Description

A very basic requirement for most registration formulars is to validate that a given registration information does not yet exist in the persistence backend (e.g. a user with an specific e-mail may only exist once).

For my current project I therefore already implemented a NotExistsValidator, which checks existence of an object with the given property value through the domain models repository.

The property and repository class can be given in the validator annotation, but since it is redundance to annotate for example an **$email** property on a **User model** with {property="email",repository="\Foo\Bar\Domain\Repository\UserRepository"} I also added an automatism to resolve those.
Therefore the **$targetClassName** and **$classPropertyName** get injected into the validator options as "__property" and "__className" in the ValidatorResolver.
Hence, this makes all Validators generally aware of the current class/property they are invoked on, which might open up a few more possibilities for validators.

Well, I'll talk straight code here to show the basic idea of the validator itself:

```
/**
 * Validator to check if an object does not exist in a repository
 *
 * @api
 */
class NotExistsValidator extends \TYPO3\FLOW3\Validation\Validator\AbstractValidator {

    /**
     * @var \TYPO3\FLOW3\Object\ObjectManager
     * @FLOW3\Inject
     */
    protected $objectManager;

    /**
     * Checks if the given value does not exist in a specified repository.
     *
     * @param mixed $value The value that should be validated
     * @param array $validationOptions Not used
     * @return void
     * @api
     */
    protected function isValid($value) {
        if (!isset($this->options['repository'])) {
            $possibleRepositoryClassName = $this->options['__className'];
            if (strpos($possibleRepositoryClassName, '\\Model\\') !== FALSE) {
                $possibleRepositoryClassName = str_replace('\\Model\\', '\\Repository\\', $possibleRepositoryClassName) .
'Repository';
                $repository = $this->objectManager->get($possibleRepositoryClassName);
            } else {
                throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The "repository" option must be
set in NotExistsValidator when not used on a Domain Model.', 1281454154);
            }
        } elseif (is_string($this->options['repository'])) {
            $repository = $this->objectManager->get($this->options['repository']);
        } elseif ($this->options['repository'] instanceof \TYPO3\FLOW3\Persistence\Repository) {
            $repository = $this->options['repository'];
        } else {
            throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The "repository" option can be only
set to string, or Repository object.', 1281454155);
        }

        if (!$repository instanceof \TYPO3\FLOW3\Persistence\RepositoryInterface) {
            throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The repository for class "' .
$this->options['__className'] . '" could not be found in NotExistsValidator.', 1281454155);
        }

        $property = $this->options['__property'];
        $countBy = 'countBy' . $property;
        $count = $repository->$countBy($value);
        if ($count > 0) {
            $this->addError('The object exists in the repository.', 1281454156);
        }
    }
}
```

What do you think? Would this make sense for an validator shipped with flow3?

Related issues:

related to TYPO3.Flow - Feature # 47191: Make (property) Validators aware of ...	**Under Review 2013-04-13**

History

#1 - 2012-08-14 10:39 - Bastian Waidelich

Hi Alexander,

interesting ideas and I think, this is a useful extension.
I'm not so sure about the className vs object heuristic as it makes the code harder to read and more error prone.
The __property & __className approach is interesting too, but this should be a separate change. Also I'd prefer them to be set with a proper setter and not through the options array (e.g. $this->getValidatedProperty() & $this->getValidatedClassName()).

I needed a unique validator the other day. here's my approach:

```
class UniqueValidator extends \TYPO3\FLOW3\Validation\Validator\AbstractValidator {

    /**
```

```
	 * @param mixed $value The value that should be validated
	 * @return void
	 * @throws \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException
	 */
	protected function isValid($value) {
		if (!isset($this->options['repository'])) {
			throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The option "repository" was not specified.', 1336499425);
		}
		if (!isset($this->options['propertyName'])) {
			throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The option "propertyName" was not specified.', 1336499429);
		}
		$repository = $this->options['repository'];
		if (!$repository instanceof \TYPO3\FLOW3\Persistence\RepositoryInterface) {
			throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The option "repository" must implement RepositoryInterface.', 1336499435);
		}
		if (isset($this->options['alwaysAllow']) && $value === $this->options['alwaysAllow']) {
			return;
		}
		$propertyName = (string)$this->options['propertyName'];
		$query = $repository->createQuery();
		$numberOfResults = $query->matching($query->equals($propertyName, $value))->count();
		if ($numberOfResults > 0) {
			$this->addError('This %s is already taken', 1336499565, array($propertyName));
		}
	}
}
```

It is a bit simpler (does not yet allow 'className') but it allows you to explicitly allow a value (e.g. if the user wants to change its profile his email must be allowed).

Another similar validator for a forgot password functionality:

```
class ExistsValidator extends \TYPO3\FLOW3\Validation\Validator\AbstractValidator {

	/**
	 * @param mixed $value The value that should be validated
	 * @return void
	 * @throws \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException
	 */
	protected function isValid($value) {
		if (!isset($this->options['repository'])) {
			throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The option "repository" was not specified.', 1340810961);
		}
		if (!isset($this->options['propertyName'])) {
			throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The option "propertyName" was not specified.', 1340810962);
		}
		$repository = $this->options['repository'];
		if (!$repository instanceof \TYPO3\FLOW3\Persistence\RepositoryInterface) {
			throw new \TYPO3\FLOW3\Validation\Exception\InvalidValidationOptionsException('The option "repository" must implement RepositoryInterface.', 1340810965);
		}
		$propertyName = (string)$this->options['propertyName'];
		$query = $repository->createQuery();
		$numberOfResults = $query->matching($query->equals($propertyName, $value))->count();
		if ($numberOfResults === 0) {
			$this->addError('This %s was not found', 1340810986, array($propertyName));
		}
	}
}
```

#2 - 2012-08-14 16:14 - Alexander Berl

Funny how you have pretty much exactly my first implementation of that validator, before I added the __property and __className magic. Only difference is that I used ValidationGroups to do the "alwaysAllow" part.

I guess adding getter/setters instead of using the options is a good idea.

With "className vs object heuristic" you mean the $possibleRepositoryClassName evaluation or the $this->options['repository'] elseif blocks?

What I'd definitely like to do, is replace the $countBy function name invocation if possible, but I'm not sure if it's a good idea to create the query in the validator, since the problem of how to count the number of objects should only be a concern of the repository IMO.

Waiting for further feedback, but this is a great start showing that my implementation is not too far off from what one would expect. Thank you!