Core - Feature # 60460

Epic # 55070 (Accepted): Workpackages
Epic # 55065 (New): WP: Overall System Performance (Backend and Frontend)
Epic # 55656 (Accepted): Optimize overall Extbase performance
Story # 55168 (Accepted): Optimize Extbase generic persistence
Task # 55169 (Under Review): Extbase: fetch child objects in one query

Status:	Accepted	**Priority:**	Could have
Author:	Felix Oertel	**Category:**	Extbase
Created:	2014-07-22	**Assigned To:**	Felix Oertel
Updated:	2015-06-15	**Due date:**	
PHP Version:			
Complexity:			
Sprint Focus:			
Subject:	Refactor lazyLoading handling in extbase		

Description

For quiet some time now I wanted to refactor the lazyLoading handling in extbase.

It stinks, that the lazyLoadingProxy does not extend the actual class, thus fataling when using correct typehints. This leads to developers allways resolving the lazyLoadingProxy which - let's face it - kind of goes against what lazyLoading was intended to do. ;-)

So we need a lazyProxy, which extends the actual class, eliminating the need to ever resolve a lazyProxy if no property is called.

Related issues:

related to Core - Task # 59917: User Repositories in internal persistence logic	**Under Review**	**2014-06-27**	**2014-07-09**
related to Core - Bug # 60913: Cannot edit lazy loaded objects	**Resolved**	**2014-08-12**	

History

#1 - 2014-07-22 15:37 - Stefano Kowalke

The new implementation introduces a LazyLoadingProxyFactory which generates a LazyProxy object which extends the concrete object. With this way proxy objects will pass type hints.

The implementation in detail:

LazyLoadingProxyFactory

The class can be found at \TYPO3\CMS\Extbase\Persistence\Generic\LazyLoadingProxyFactory. To retrieve a lazy proxy object call the get() method of the class, which expects three parameters:

1. The classname or object to create the proxy from
2. The parent object
3. The property
4. The value of the field

$this->lazyProxyFactory->getProxy($propertyMetaData['type'], $parentObject, $propertyName, $fieldValue);

The generated proxy class is cached and you will find it at typo3temp/Cache/Code/extbase_lazyproxyobject_storage/*LazyProxy.php

Example

Lets use the blog_example as … well … as an example:
At some point the DataMapper calls $this->lazyProxyFactory->getProxy(…) with this values:
1. string $className: ExtbaseTeam\BlogExample\Domain\Model\Administrator
2. object $parentObject: ExtbaseTeam\BlogExample\Domain\Model\Blog
3. string $propertyName: administrator

4. string $fieldValue: 351

Here is the code of the generated AdministratorLazyProxy.php:

```
<?php
namespace ExtbaseTeam\BlogExample\Domain\Model;
use TYPO3\CMS\Extbase\Persistence\QueryResultInterface;

/**
 * This class is generated by the LazyProxyFactory of Extbase. Don't change the file by yourself!
 */
class AdministratorLazyProxy extends \ExtbaseTeam\BlogExample\Domain\Model\Administrator implements
\TYPO3\CMS\Extbase\Persistence\Generic\LoadingStrategyInterface {
    private $parentObject;
    private $propertyName;
    private $fieldValue;
    private $parentQueryResult;

    public function __construct($parentObject, $propertyName, $fieldValue) {
        $this->parentObject = $parentObject;
        $this->propertyName = $propertyName;
        $this->fieldValue = $fieldValue;
    }

    public function setUsername($username) {
        $this->parentQueryResult->fetchLazyObjects($this->propertyName);
        return parent::setUsername($username);
    }

    public function getUsername() {
        $this->parentQueryResult->fetchLazyObjects($this->propertyName);
        return parent::getUsername();
    }

    ...

    /**
     * @param QueryResultInterface $queryResult
     *
     * @return void
     */
    public function setParentQueryResult(QueryResultInterface $queryResult) {
        $this->parentQueryResult = $queryResult;
    }

    /**
     * Returns the parentObject so we can populate the proxy.
     *
     * @return object
     */
    public function _getParentObject() {
        return $this->parentObject;
    }

    /**
     * Returns the fieldValue so we can fetch multiple LazyObjects in one query.
     *
     * @return mixed
     */
    public function _getFieldValue() {
        return $this->fieldValue;
    }
}
```

The proxy class overrides all public methods from his parent with his own variant of the methods. This just just add a call to the methods whereby the proxy is able to resolve himself. At the end it will return the requested value from the concrete object.

#2 - 2014-08-15 09:50 - Mathias Brodala

Can you push a changeset to Gerrit to get this ball rolling again?

#3 - 2014-08-18 08:59 - Felix Oertel

Hey Mathias,

sorry, the patch was with me for some polish ... an I was on vacation. ;-) But it's ready now and I am gonna push it today.

#4 - 2014-12-23 19:46 - Mathias Schreiber

- Target version changed from 7.0 to 7.1 (Cleanup)

#5 - 2015-03-13 09:55 - Stephan Großberndt

The patch is in https://review.typo3.org/#/c/32286/

#6 - 2015-06-05 18:40 - Christian Kuhn

the pending patch was abandoned.

#7 - 2015-06-15 17:07 - Benjamin Mack

- Target version changed from 7.1 (Cleanup) to 7.4 (Backend)