TYPO3.Flow - Task # 55957

Status:	New	**Priority:**	Should have
Author:	Christopher Hlubek	**Category:**	AOP
Created:	2014-02-13	**Assigned To:**	
Updated:	2014-02-28	**Due date:**	
Sprint:			
PHP Version:			
Has patch:	No		
Complexity:			
Subject:	RFC: Optimize AOP proxies		

Description

Our current approach of applying AOP in generated proxies has some downsides:

- It's hard to follow the dynamic calls during debugging (Flow_Aop_Proxy_invokeJoinPoint, call_user_func_array, $adviceObject->$methodName($joinPoint))
- Profiling information contains a lot of these dynamic calls where it's again hard to follow parent / child call relations
- The creation of all advices for all methods in the constructor has some overhead
- The *Advice objects that are used to evaluate the advices dynamically add an additional overhead and nesting
- The proxy code is hard to understand, it's not obvious what a proxy method will execute

Idea:

- Unroll the advices in the proxied method, this is simple for everything but Around
- Use a direct call to the aspect method for easier to debug code
- Use a direct call to the original method by using a closure instead of the dynamic invocation with Flow_Aop_Proxy_invokeJoinPoint

Sketch:

```
<?php

class TargetClass01 extends TargetClass01_Original implements \TYPO3\Flow\Object\Proxy\ProxyInterface {

    /**
     * Autogenerated Proxy Method
     */
    public function __construct() {

        if (isset($this->Flow_Aop_Proxy_methodIsInAdviceMode['__construct'])) {
            parent::__construct();
        } else {
            $this->Flow_Aop_Proxy_methodIsInAdviceMode['__construct'] = TRUE;
            try {
                $methodArguments = array();

                // The advice chain is only needed for an Around advice, before advices could be directly placed here
                // The advice chain is composed of a list of closures that actually call the method / advices to have an explicit call instead of call_user_func_array
                // (we could cache the advice chain instances per method if we can measure a performance improvement)
                $adviceChain = new LightweightAdviceChain(function($joinPoint) {
```

```
                // Needs PHP 5.4
                return parent::__construct();
            });

            $joinPoint = new \TYPO3\Flow\Aop\JoinPoint($this, 'TYPO3\Flow\Tests\Unit\Aop\Fixtures\TargetClass01',
'__construct', $methodArguments, $adviceChain);

            $aspect =
\TYPO3\Flow\Core\Bootstrap::$staticObjectManager->get('TYPO3\Flow\Tests\Functional\Aop\Fixtures\BaseFunctionalityTestingAspect');
gAspect');

            // The advices are invoked explicitly for easier debugging and profiling
            $result = $aspect->lousyConstructorAdvice($joinPoint);

        } catch (\Exception $e) {
            unset($this->Flow_Aop_Proxy_methodIsInAdviceMode['__construct']);
            throw $e;
        }
        unset($this->Flow_Aop_Proxy_methodIsInAdviceMode['__construct']);
        return;
    }

    if (get_class($this) === 'TYPO3\Flow\Tests\Unit\Aop\Fixtures\TargetClass01') {
        $this->initializeObject(1);
    }
  }
}
```

History