# Источники

1. Tutorialspoint - Spring

# AOP with Spring Framework(1)

***Таблицы: Описание понятий AOP и типы Advise***

Поясняющие примеры оказывается могут быть спрятаны в разделе

|  |  |
| --- | --- |
| **Approach** | **Description** |
| [XML Schema based](http://www.tutorialspoint.com/spring/schema_based_aop_appoach.htm) | Aspects are implemented using regular classes along with XML based configuration. |
| [@AspectJ based](http://www.tutorialspoint.com/spring/aspectj_based_aop_appoach.htm) | @AspectJ refers to a style of declaring aspects as regular Java classes annotated with Java 5 annotations. |

## XML Schema Based AOP with Spring

Здесь начались проблемы.

Во первых пришлось скачать библиотеки **AspectJ**. После скачал apache common-logging библиотеки, но почитав содержание stackTrace понял, что проблема не в этом. А проблема оказалась в том, что отсутствовал класс ***org.aopalliance.aop.Advice .*** Как оказалось, что с 2011 года соответствующая библиотека исключена из Spring. Скачал <http://mvnrepository.com/artifact/aopalliance/aopalliance/1.0>, получил результат.

## @AspectJ Based AOP with Spring

Использую тот же проект: второй конфигурационный файл и пакет.

## 00 Tutorialspoint

Отработаны темы [Scope](http://www.tutorialspoint.com/spring/spring_bean_scopes.htm) и [Post](http://www.tutorialspoint.com/spring/spring_bean_post_processors.htm)Processing

|  |  |
| --- | --- |
| **bean.xml** |  |
| <bean id="helloWorld" class="tutorialspoint.spring.HelloWorld"  init-method="init" destroy-method="destroy" scope="prototype" >  <!-- -->    <property name="message" value="Hello World!" />  </bean>  <bean class="tutorialspoint.spring.InitHelloWorld" /> |  |
| **HelloWorld** |  |
| private String message;  public void setMessage(String message){  this.message = message;  }  public void getMessage(){  System.out.println("Your Message : " + message);  }    public void init(){  System.out.println("Bean is going through init.");  }  public void destroy(){  System.out.println("Bean will destroy now.");  } |  |
| **InitHelloWorld** |  |
| import *org.springframework.beans.factory.config.BeanPostProcessor*;  import *org.springframework.beans.BeansException*;  …  public Object postProcessBeforeInitialization(Object bean, String beanName) throws BeansException {  System.out.println("BeforeInitialization : " + beanName);  return bean; // you can return any other object as well  }  public Object postProcessAfterInitialization(Object bean, String beanName) throws BeansException {  System.out.println("AfterInitialization : " + beanName);  return bean; // you can return any other object as well  } |  |
| **MainApp** |  |
| AbstractApplicationContext context  = new ClassPathXmlApplicationContext("Beans.xml");  HelloWorld objA = (HelloWorld) context.getBean("helloWorld");  objA.getMessage();  objA.setMessage("I'm object A");    HelloWorld objB = (HelloWorld) context.getBean("helloWorld");  objB.setMessage("I'm object B");  objB.getMessage();  objA.getMessage(); | Работает по разному в зависимости от scope. Если prototype то каждый раз bean инициализируется по новой |
| *context.registerShutdownHook();* | Уничтожает bean, но только если scope=singleton (default). Если scope=prototype то не срабатывает |