Spring Aop pointcut expression

Теги: Spring AOP Pointcut

В Spring AOP обычно необходимо использовать язык выражений pointcut AspectJ для определения pointcut. Важно то, что в Spring поддерживается только подмножество индикаторов pointcut AspectJ.

AspectJ Pointcut индикатор поддерживается Spring

Индикатор AspectJ	описание
args()	Метод выполнения, который ограничивает параметры сопоставления точки подключения типом выполнения
@args()	Метод выполнения, который ограничивает пометки параметров соответствия точки подключения аннотациями выполнения
execution()	Соответствующий метод выполнения точки подключения
this()	Ограничьте ссылочный тип Бина точки соединения, соответствующей прокси-серверу AOP, указанным типом Бина
target()	Ограничить точки подключения от сопоставления целевых объектов до классов указанных типов
@target()	Ограничить точки подключения от сопоставления целевого объекта с указанным классом аннотаций
within()	Ограничить точки подключения, чтобы соответствовать указанному типу
@within()	Ограничить точки подключения, чтобы соответствовать указанному типу аннотации
@annotation	Ограничить сопоставление точек подключения с указанными аннотациями

Обычно в Spring AOP используются:

```
execution(modifiers-pattern? ret-type-pattern declaring-type-pattern? name-pattern(param-pattern)
          throws-pattern?)
```

Сопоставить все

```
execution("* *.*(..)")
```

Сопоставить все методы, начинающиеся с set

```
execution("* *.set*(..))
```

Сопоставить все методы в указанном пакете

```
execution("* com.david.biz.service.impl.*(..))
```

Сопоставить все методы в указанном пакете и его подпакетах

```
execution("* com.david..*(..)")
```

Соответствует указанному пакету, а его тип параметра подпакета - String

```
execution("* com.david..*(java.lang.String))
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:p="http://www.springframework.org/schema/p" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:aop="http://www.springframework.org/schema/aop" xmlns:jee="http://www.springframework.org/schema/jee"
        xmlns:task="http://www.springframework.org/schema/task"
        xsi:schemaLocation="
                http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                http://www.springframework.org/schema/context
                http://www.springframework.org/schema/context/spring-context-3.0.xsd
                http://www.springframework.org/schema/aop
                http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
                http://www.springframework.org/schema/tx
                http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
                http://www.springframework.org/schema/jee
                http://www.springframework.org/schema/jee/spring-jee-3.0.xsd
                http://www.springframework.org/schema/task
        http://www.springframework.org/schema/task/spring-task-3.1.xsd
                ">
        <context:component-scan base-package="com.david.*"/>
        <aop:aspectj-autoproxy />
        <context:property-placeholder location="classpath:META-INF/config.properties" />
         <! - Определить источник данных ->
        <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
                destroy-method="close">
                <property name="driverClass" value="${jdbc.ams.driver}" />
```

```
                <property name="jdbcUrl" value="${jdbc.ams.url}" />
                <property name="user" value="${jdbc.ams.username}" />
                <property name="password" value="${jdbc.ams.password}" />
                <property name="initialPoolSize" value="${initialSize}" />
                <property name="minPoolSize" value="${minPoolSize}" />
                <property name="maxPoolSize" value="${maxActive}" />
                <property name="acquireIncrement" value="${acquireIncrement}" />
                <property name="maxIdleTime" value="${maxIdleTime}" />
        </bean>

         <! - Определить класс шаблона jdbc ->
        <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
                <property name="dataSource" ref="dataSource" />
        </bean>

        <bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
                <property name="dataSource" ref="dataSource" />
                <property name="configLocation" value="classpath:META-INF/sqlmap/sqlmap.xml" />
        </bean>

        <tx:advice id="txAdvice" transaction-manager="transactionManager">
                <tx:attributes>
                        <tx:method name="addBook" propagation="REQUIRED" />
                        <tx:method name="addUserBook" propagation="MANDATORY" />
                        <tx:method name="deleteBook" propagation="REQUIRES_NEW" />
                        <tx:method name="addNewBook" propagation="NEVER" />
                        <tx:method name="addUser" propagation="NESTED" />
                </tx:attributes>
        </tx:advice>
        <bean id="transactionManager"
                class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
                <property name="dataSource" ref="dataSource" />
        </bean>

</beans>
```

```
@Service("bookService")
public class BookServiceImpl implements BookService {

    private static final Logger logger      = LogManager.getLogger(BookServiceImpl.class);
    public static final String  ADD_BOOK    = "insert into t_book(id,name) values(1,'duck-j2ee')";

    public static final String  DELETE_BOOK = "delete from  t_book where id=1";

    private JdbcTemplate        jdbcTemplate;
    @Autowired
    private BookDao             bookDao;

    public void addBook() throws Exception {
        Book book = new Book();
        book.setName("ibatis");
        book.setPrice(11);
        bookDao.insert(book);
                 генерировать новое исключение UnRollbackException («Проверено исключение, откат не будет»);
    }

    public void deleteBook(int id) {
        try {
            bookDao.deleteById(id);
        } catch (SQLException e) {
            logger.error("", e);
        }
    }

    @LoggingRequired
    public void addNewBook(String name, int price) {
        try {
            Book book = new Book();
            book.setName(name);
            book.setPrice(price);
            bookDao.insert(book);
            List<Book> lists = bookDao.selectAll();
            System.out.println(lists);
        } catch (SQLException e) {
            logger.error("", e);
        }
    }

    public void addUserBook() {
        jdbcTemplate.execute("insert into t_book(id,name) values(3,'UserBook')");
    }

    /**
     * Setter method for property <tt>jdbcTemplate</tt>.
     *
     * @param jdbcTemplate value to be assigned to property jdbcTemplate
     */
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    /**
     * @see com.david.biz.service.BookService#queryAll()
     */
```

```
    public List<Book> queryAll() {
        try {
            return bookDao.selectAll();
        } catch (SQLException e) {
            logger.error("", e);
        }
        return null;
    }

}
```

```
/**
 * execution (modifiers-pattern? ret-type-pattern declaring-type-pattern? name-pattern (param-pattern)
          throws-pattern?)
   * arg () Метод выполнения, который ограничивает параметр сопоставления точки подключения указанным типом
   * @ args () Ограничить выполнение параметров сопоставления точек подключения, отмеченных аннотациями выполнения
   * execute () используется для соответствия методу выполнения точки подключения
   * this () ограничивает точку подключения, соответствующую ссылке Bean-адреса прокси-сервера AOP, в качестве класса типа
   * target () Ограничить точку подключения, чтобы сопоставить целевой объект с указанным типом класса
   * @ target () Ограничить точки подключения, чтобы соответствовать конкретным объектам выполнения, эти объекты должны им
   * @ annotation () ограничивает сопоставление точек подключения с указанными аннотациями
 *
 *
 *
 * @author zhangwei_david
   * @version $ Id: LogAspect.java, v 0.1 29 ноября 2014 г. 13:10:13 zhangwei_david Exp $
 */
@Component
@Aspect
public class LogAspect {
    private static final Logger logger = LogManager.getLogger(LogAspect.class);

    /**
     * Соответствующим параметром является любой тип, любое число и метод в пакете com, david.biz или подпакете.
     */
    @Pointcut("args(..)&&within(com.david.biz..*)")
    public void arg() {

    }

    @Pointcut("@args(com.david.aop.LoggingRequired)")
    public void annotationArgs() {

    }

    @Pointcut("@annotation(com.david.aop.LoggingRequired)")
    public void logRequiredPointcut() {

    }

    @Pointcut("args(java.lang.String,*)")
    public void argsWithString() {

    }

    @Pointcut("target(com.david.biz.service.impl.BookServiceImpl)")
    public void targetPointcut() {

    }

    @Pointcut("@target(org.springframework.stereotype.Service)")
    public void targetAnnotation() {

    }

    //    @Around("execution(* org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter.handle(..))"
    //    public Object aa(ProceedingJoinPoint pjp) throws Throwable {
    //        try {
    //            Object retVal = pjp.proceed();
    //            System.out.println(retVal);
    //            return retVal;
    //        } catch (Exception e) {
         // System.out.println ("Исключение");
    //            return null;
    //        }
    //    }
    @Before(value = "logRequiredPointcut()")
    public void before(JoinPoint joinPoint) {
        LogUtils.info(logger,
                      «Выражение точки подключения @annotation (com.david.aop.LoggingRequired) -method = {0} было посе
            joinPoint.getSignature().getName());
    }

    @Before(value = "arg()")
    public void beforeArg(JoinPoint joinPoint) {
        LogUtils.info(logger,
                      «Выражение точки соединения: args (..) && в (com.david.biz .. *) метод = {0}, args = {1}, target
            joinPoint.getSignature().getName(), ToStringBuilder.reflectionToString(
                joinPoint.getArgs(), ToStringStyle.SHORT_PREFIX_STYLE), joinPoint.getTarget()
                .getClass().getName());
    }

    @Before(value = "argsWithString()")
```

```
    public void beforeArgWithString(JoinPoint joinPoint) {
                 LogUtils.info (logger, «Выражение точки соединения: метод args (java.lang.String, *) = {0}, args = {1}, 
            joinPoint.getSignature().getName(), ToStringBuilder.reflectionToString(
                joinPoint.getArgs(), ToStringStyle.SHORT_PREFIX_STYLE), joinPoint.getTarget()
                .getClass().getName());
    }

    @Before(value = "annotationArgs()")
    public void beforeAnnotationArgs(JoinPoint joinPoint) {
        LogUtils
            .info(
                logger,
                                «Выражение точки соединения: @args (com.david.annotation.validate.Length, *) method = {0
                joinPoint.getSignature().getName(), ToStringBuilder.reflectionToString(
                    joinPoint.getArgs(), ToStringStyle.SHORT_PREFIX_STYLE), joinPoint.getTarget()
                    .getClass().getName());
    }

    @Before(value = "targetPointcut()")
    public void beforeTarget(JoinPoint joinPoint) {
        LogUtils
        .info(
            logger,
                        «Выражение точки подключения: метод target (com.david.biz.service.impl.BookServiceImpl) = {0}, a
            joinPoint.getSignature().getName(), ToStringBuilder.reflectionToString(
                joinPoint.getArgs(), ToStringStyle.SHORT_PREFIX_STYLE), joinPoint.getTarget()
                .getClass().getName());
    }

    @Before(value = " targetAnnotation()")
    public void beforeTargetAnnotation(JoinPoint joinPoint) {
        LogUtils
            .info(
                logger,
                                «Выражение точки подключения: метод @target (org.springframework.stereotype.Service) = {
                joinPoint.getSignature().getName(), ToStringBuilder.reflectionToString(
                    joinPoint.getArgs(), ToStringStyle.SHORT_PREFIX_STYLE), joinPoint.getTarget()
                    .getClass().getName());
    }
}
```

```
/**
 *
 * @author zhangwei_david
   * @version $ Id: T.java, v 0.1 1 декабря 2014 г. 9:35:44 zhangwei_david Exp $
 */
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "file:H:/workspace4study/WebApp/src/main/webapp/WEB-INF/applicationContext.xml")
public class BookServiceTest {

    @Autowired
    private BookService bookService;

    @Test
    public void testB() {
        bookService.addNewBook("Junit  Test", 1000);
    }

}
```

```
2014-12-01 11:14:39 [main: 1577] - [INFO] Выражение точки подключения @annotation (com.david.aop.LoggingRequired) -method
 2014-12-01 11:14:39 [main: 1587] - [INFO] Выражение точки подключения: args (..) && в (com.david.biz .. *) метод = addNe
 2014-12-01 11:14:39 [main: 1588] - [INFO] Выражение точки подключения: args (java.lang.String, *) method = addNewBook, a
 2014-12-01 11:14:39 [main: 1588] - [INFO] Выражение точки подключения: target (com.david.biz.service.impl.BookServiceImp
 2014-12-01 11:14:39 [main: 1589] - [INFO] Выражение точки подключения: @target (org.springframework.stereotype.Service) 
 2014-12-01 11:14:39 [main: 1589] - [INFO] Выражение точки подключения: args (..) && в (com.david.biz .. *) метод = inser
 2014-12-01 11:14:39 [main: 1590] - [INFO] Выражение точки подключения: @args (com.david.annotation.validate.Length, *) me
 2014-12-01 11:14:39 [main: 1591] - [INFO] Выражение точки подключения: args (java.lang.String, *) method = insert, args =
```