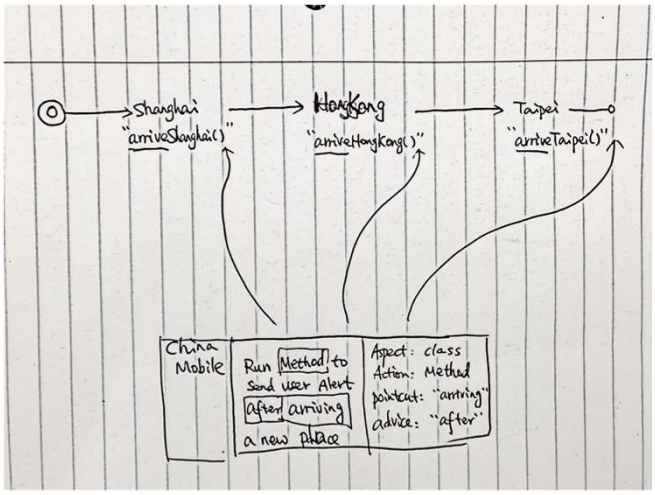
Aspect Programming

Andy

1

AOP: Aspect Oriented Programming



- Aspect: class to define all aop self methods
- Pointcut: expression to find all main application methods to insert AOP
- Advice: before/after/around/after throwing – when to execute aop methods when the application pointcut is found

2

Annotation (annotation + class)

```
@Aspect
public class ConcurrentOperationExecutor implements Ordered {

    private static final int DEFAULT_MAX_RETRIES = 2;

    private int maxRetries = DEFAULT_MAX_RETRIES;
    private int order = 1;

    public void setMaxRetries(int maxRetries) {
        this.maxRetries = maxRetries;
    }

    public int getOrder() {
        return this.order;
    }

    public void setOrder(int order) {
        this.order = order;
    }

    @Around("com.xyz.myapp.SystemArchitecture.businessService()")
    public Object doConcurrentOperation(ProceedingJoinPoint pjp) throws Throwable {
        int numAttempts = 0;
        PessimisticLockingFailureException lockFailureException;
        do {
            numAttempts++;
            try {
                return pjp.proceed();
            }
            catch(PessimisticLockingFailureException ex) {
                lockFailureException = ex;
            }
        } while(numAttempts <= this.maxRetries);
        throw lockFailureException;
    }

}
```

3

XML (config+class)

```
<aop:config>

    <aop:aspect id="concurrentOperationRetry" ref="concurrentOperationExecutor">

        <aop:pointcut id="idempotentOperation"
            expression="execution(* com.xyz.myapp.service.*.*(..))"/>

        <aop:around
            pointcut-ref="idempotentOperation"
            method="doConcurrentOperation"/>

    </aop:aspect>

</aop:config>

<bean id="concurrentOperationExecutor"
    class="com.xyz.myapp.service.impl.ConcurrentOperationExecutor">
        <property name="maxRetries" value="3"/>
        <property name="order" value="100"/>
</bean>
```

```
public class ConcurrentOperationExecutor implements Ordered {

    private static final int DEFAULT_MAX_RETRIES = 2;

    private int maxRetries = DEFAULT_MAX_RETRIES;
    private int order = 1;

    public void setMaxRetries(int maxRetries) {
        this.maxRetries = maxRetries;
    }

    public int getOrder() {
        return this.order;
    }

    public void setOrder(int order) {
        this.order = order;
    }


    public Object doConcurrentOperation(ProceedingJoinPoint pjp) throws Throwable {
        int numAttempts = 0;
        PessimisticLockingFailureException lockFailureException;
        do {
            numAttempts++;
            try {
                return pjp.proceed();
            }
            catch(PessimisticLockingFailureException ex) {
                lockFailureException = ex;
            }
        } while(numAttempts <= this.maxRetries);
        throw lockFailureException;
    }

}
```

4

@ExceptionHandler (intercept a particular exception inside one class)

```java
@Controller
public class MainController {

    @RequestMapping(value = "/{type:.+}", method = RequestMethod.GET)
    public ModelAndView getPages(@PathVariable("type") String type)
        throws Exception {

      if ("error".equals(type)) {
        // go handleCustomException
        throw new CustomGenericException("E888", "This is custom message");
      } else if ("io-error".equals(type)) {
        // go handleAllException
        throw new IOException();
      } else {
        return new ModelAndView("index").addObject("msg", type);
      }

    }

    @ExceptionHandler(CustomGenericException.class)
    public ModelAndView handleCustomException(CustomGenericException ex) {

        ModelAndView model = new ModelAndView("error/generic_error");
        model.addObject("errCode", ex.getErrCode());
        model.addObject("errMsg", ex.getErrMsg());

        return model;

    }

    @ExceptionHandler(Exception.class)
    public ModelAndView handleAllException(Exception ex) {

        ModelAndView model = new ModelAndView("error/generic_error");
        model.addObject("errMsg", "this is Exception.class");

        return model;

    }

}
```

5

@ExceptionHandler (intercept a particular exception in whole application)

```java
@ControllerAdvice
public class GlobalExceptionController {

    @ExceptionHandler(CustomGenericException.class)
    public ModelAndView handleCustomException(CustomGenericException ex) {

        ModelAndView model = new ModelAndView("error/generic_error");
        model.addObject("errCode", ex.getErrCode());
        model.addObject("errMsg", ex.getErrMsg());

        return model;

    }

    @ExceptionHandler(Exception.class)
    public ModelAndView handleAllException(Exception ex) {

        ModelAndView model = new ModelAndView("error/generic_error");
        model.addObject("errMsg", "this is Exception.class");

        return model;

    }

}
```

6