ENTERPRISE DEVELOPMENT SERVICES DIRECT FROM THE CREATORS OF SPRING FRAMEWORK

INTERFACE21

AOP in the Enterprise

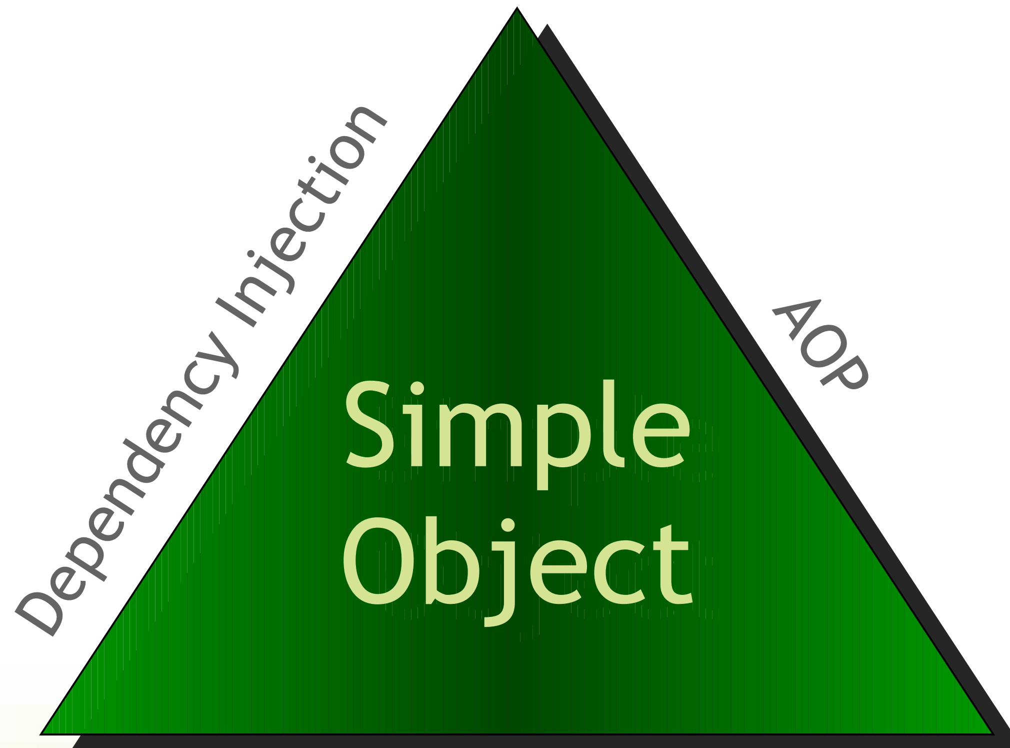
Adrian Colyer
CTO
Interface21

Spring

Agenda

- Where does AOP fit?
- Matching requirements to implementation
- AOP in Spring 2.5

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.



Portable Service Abstractions

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

the vocabulary of enterprise applications

business service

service layer

repository

dao

controller

web layer

data access layer

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

INTERFACE21

Requirements

- the service layer should be transactional
- when a Hibernate dao operation fails the exception should be translated
- a business service that fails with a concurrency related failure can be retried
- service layer objects should not call the web layer

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

INTERFACE21

It would be simpler…

and more *powerful*

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

if we could use these

~~terms~~ abstractions

directly in the

implementation

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

terms -> abstractions

```
@Aspect
public class SystemArchitecture {

  @Pointcut("within(a.b.c.service..*")
  public void inServiceLayer() {}

  @Pointcut("within(a.b.c.dao..*")
  public void inDataAccessLayer() {}

  @Pointcut("execution(* a.b.c.service.*.*(..))")
  public void businessService() {}


  …
}
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Requirements

- **the service layer should be transactional**
  - when a Hibernate dao operation fails the exception should be translated
  - a business service that fails with a concurrency related failure can be retried
  - service layer objects should not call the web layer

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Transactional service layer…

```
<aop:config>
  <aop:advisor
     pointcut="SystemArchitecture.businessService()"
     advice-ref="tx-demarcation"/>
</aop:config>
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

INTERFACE21

Transaction metadata

```
<tx:advice id="tx-demarcation">
  <method name="*"
    propagation="REQUIRED"
    isolation="DEFAULT"/>
</tx:advice>
```

- Gives us TX-REQUIRED semantics for the service layer

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

Transactional annotation

```
/**
 * default to required, read-write for all
 *  operations
 */
@Transactional
public class AccountService {
  /** this one is read-only… */
  @Transactional(readOnly=true)
  public Account getAccount(AccountNum accNo) {
   …
  }
  …
}
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or
distributing without expressed written permission is prohibited.

Spring

Transactional annotation

- Now the configuration just becomes…

```
<!– tell Spring to perform transaction demarcation on
   bean operations based on @Transactional annotations
   -->
<tx:annotation-driven/>
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

INTERFACE21

Requirements

- ✔ the service layer should be transactional
- ➔ **when a Hibernate dao operation fails the exception should be translated**
- a business service that fails with a concurrency related failure can be retried
- service layer objects should not call the web layer

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

INTERFACE21

Scenario…

- You have your own data access layer written using Hibernate 3
  - not using the Spring HibernateTemplate
- In the service layer, you want to insulate yourself from Hibernate exceptions, and take advantage of Spring's fine-grained DataAccessException hierarchy
- After throwing a hibernate exception from a data access operation, convert it into a DataAccessException…

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

Step 1: Define the abstraction

```
@Aspect
public class SystemArchitecture {

  …

  @Pointcut("execution(* a.b.c.dao.*.*(..))")
  public void dataAccessOperation() {}

  …
}
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or
distributing without expressed written permission is prohibited.

"After throwing a hibernate exception from a data access operation, convert it into a DataAccessException…"

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

Step 2: use the abstraction

```
@AfterThrowing(
  throwing="hibernateEx",
  pointcut="SystemArchitecture.dataAccessOperation()" )
public void rethrowAsDataAccessException(
  HibernateException hibernateEx) {
  // convert exception and rethrow…
}
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

INTERFACE21

Where does advice live?

- Advice is declared in an aspect
- Aspects are like classes
  - instances
  - state (fields)
  - behaviour (methods)
- Aspects can also have
  - pointcuts
  - advice
  - and a few other things…

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or
distributing without expressed written permission is prohibited.

Spring

Aspect

```
@Aspect
public class HibernateExceptionTranslator {
     // …

    @AfterThrowing(
      throwing="hibernateEx",
      pointcut="SystemArchitecture.dataAccessOperation()"
    )
    public void rethrowAsDataAccessException(
      HibernateException hibernateEx) {
      // convert exception and rethrow…
    }
}
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or
distributing without expressed written permission is prohibited.

Spring

Step 3: Configuration

```
<aop:aspectj-autoproxy/>

<context:component-scan>
  <context:include-filter type="annotation"
      expression="org.aspectj.lang.annotation.Aspect"/>
</context:component-scan>
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Schema alternative

- For JDK 1.4 and below
- The exact same aspect can be declared in Spring XML, backed by a plain Java class

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Schema-based configuration

```
<aop:config>

  <aop:aspect ref="hibernateExceptionTranslator">
    <aop:after-throwing
        throwing="hibernateEx"
        pointcut="SystemArchitecture.dataAccessOperation()"
        method="rethrowAsDataAccessException"/>
  </aop:aspect>

</aop:config>
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

Bean Implementation

```
public class HibernateExceptionTranslator {
  private HibernateTemplate hibernateTemplate;

  public void setHibernateTemplate(
    HibernateTemplate aTemplate){
    this.hibernateTemplate = aTemplate;
  }

  public void rethrowAsDataAccessException(
       HibernateException hibernateEx) {
    throw this.hibernateTemplate
        .convertHibernateAccessException(hibernateEx);
  }
}
```

parameter bound
in pcut expression

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or
distributing without expressed written permission is prohibited.

@Repository

- This exception translation is available "out of the box" in Spring 2.0 and above
- Simply annotate repository / dao objects with @Repository
- Define the exception translation bean in your configuration
  - PersistenceExceptionTranslationPostProcessor

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

INTERFACE21

Requirements

- the service layer should be transactional
- when a Hibernate dao operation fails the exception should be translated
- **a business service that fails with a deadlock loser failure can be retried**
- service layer objects should not call the web layer

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

Retry

- One subtype of DataAccessException is…
  - DeadlockLoserDataAccessException
- Deadlock failures are potentially recoverable
  - If the operation is idempotent, we can retry it*
- We need a DeadlockLoserRetry aspect…

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Deadlock Loser Recovery

```
@Aspect
public class DeadlockLoserRetry {
  private static final int DEFAULT_MAX_ATTEMPTS = 3;
  private int maxAttempts = DEFAULT_MAX_ATTEMPTS;

  /** configurable via dependency injection */
  public void setMaxAttempts(int newMax) {
    this.maxAttempts = newMax;
  }


 …
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or
distributing without expressed written permission is prohibited.

Deadlock Loser Recovery

```
@Around("idempotentOperation()")
public Object retryDeadlockLosers(ProceedingJoinPoint pjp)
throws Throwable {
  int attempts = 0;
  DeadlockLoserDataAccessException loserEx = null;
  while (attempts++ < maxAttempts) {
    try {
      return pjp.proceed();
    }
    catch (DeadlockLoserDataAccessException ex) {
      loserEx = ex;
    }
  }
  throw loserEx;
}
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Schema-based equivalent

```
<aop:config>

  <aop:aspect ref="deadlockLoserRetry">
    <aop:pointcut id="idempotentOperation"
        expression= "SystemArchitecture.businessService()"/>

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

</aop:config>
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

Schema-based Equivalent

```
<bean id="deadlockLoserRetry"
   class="DeadlockLoserRetry">
   <property name="maxAttempts" value="2"/>
</bean>
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Recap:

- Created an abstraction: idempotentOperation
- Used around advice to retry failing idempotentOperations
- Packaged in a ConcurrentOperationExecutor aspect
- Configured using Spring

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Idempotent operations

- It would be nice if all of our service layer operations were idempotent
- But what if some of them aren't?
- We'd need a way to identify and retry only the idempotent subset…

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Idempotent operations

```
@Retention(RetentionPolicy.RUNTIME)
public @interface Idempotent {}
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Idempotent operations

- Just update the abstraction (pointcut expression)…

```
<aop:pointcut id="idempotentOperation"
    expression=
        "SystemArchitecture.businessService()
         and @annotation(Idempotent)"/>
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Requirements

- the service layer should be transactional
- when a Hibernate dao operation fails the exception should be translated
- a business service that fails with a concurrency related failure can be retried
- **service layer objects should not call the web layer**

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Permitted component interactions

```
/** … */
public aspect SystemArchitecture {
 …
 /*
  * no other module should depend on the
  * web tier
  */
 declare warning
   : callToWebTier() && !inWebTier()
   : "no external dependencies on web tier";

  …
}
```

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Demo

- Hibernate usage guidelines…

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

ENTERPRISE DEVELOPMENT SERVICES DIRECT FROM THE CREATORS OF SPRING FRAMEWORK

INTERFACE21

AOP in Spring 2.5

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

Spring 2.5 AOP

- Aspects are defined in Spring configuration file
  - supports both XML based definition
  - and @AspectJ aspects
- XML defined aspects are backed by regular classes

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring 2.5 and AspectJ

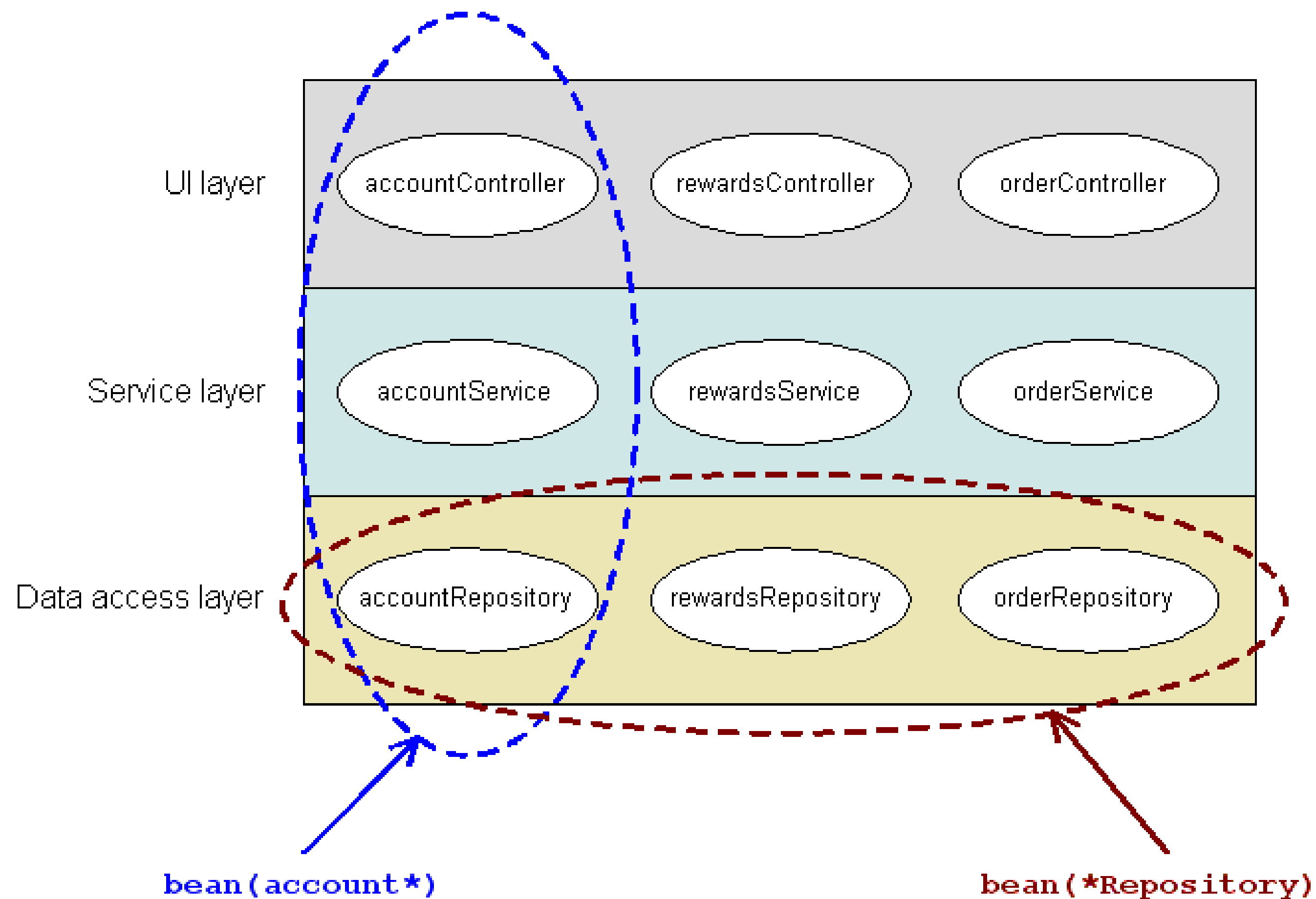
- Spring and AspectJ are still distinct projects
- Spring just uses the AspectJ pointcut parsing and matching APIs
  - using AspectJ as a library, not as a weaving engine
- Gives the same syntax and semantics across Spring AOP and AspectJ
  - perfect if you are going to use both
  - or start out with Spring AOP, and then want to introduce AspectJ at some point

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

New in Spring 2.5

- bean()



Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

New in Spring 2.5

- <context:load-time-weaver/>
  - aspectj-weaving="on | off | autodetect"

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

Summary

- We want to implement enterprise requirements in as simple and straightforward a manner as possible
  - use the appropriate implementation "vocabulary"
- AOP provides the necessary abstractions
- AspectJ and Spring AOP are the leading AOP implementations
  - tightly integrated
  - Can use together or independently

Copyright 2004-2006, Interface21 Ltd. Copying, publishing, or distributing without expressed written permission is prohibited.

Questions?