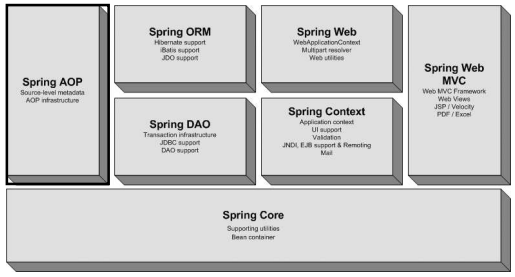
CS520 Web Programming

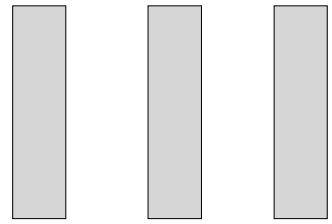
Spring – Aspect Oriented Programming

Chengyu Sun
California State University, Los Angeles

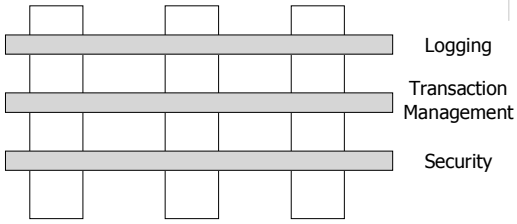
Spring Framework



Concerns



Cross-Cutting Concerns



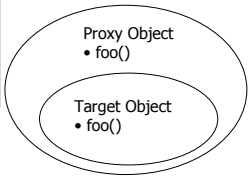
An Example of Transaction Rollback

```
catch( SQLException e ) {
   System.err.println( e.getMessage() );
   if( c != null ) {
      try {
         c.rollback();
       } catch( SQLException ex ) {
         System.err.println( ex.getMessage() );
      }
   }
}
```

Aspect Oriented Programming

- Separate out cross-cutting concern code into their own classes or modules, called aspects.
- Example: logging

How It Works




```
proxy.foo()
{
  // before target.foo() is called
  …

  // call target.foo()
  value = target.foo();

  // after target.foo() is called
  …

  return value;
}
```

AOP Terminology …

- Aspect
- Proxy and Target
- Advice: the implementation of an aspect
- Join point: a point in the execution of the application where the aspect can be plugged in
- Pointcut: determines what advice(s) should be applied at what join points

… AOP Terminology

- Introduction: adding new methods and/or fields to existing classes
- Weaving
  - Compile time
  - Class load time
  - Runtime

Spring AOP

- Advices are written in Java
- Pointcuts are defined in XML configuration file
- Supports only method join points
- Aspects are woven in at runtime
- Advisor = Aspects + Pointcuts

Advice Types

Type	Interface
Around	org.aopalliance.intercept.MethodInterceptor
Before	org.springframework.aop.BeforeAdvice
After	org.springframework.aop.AfterReturningAdvice
Throws	org.springframework.aop.ThrowsAdvice

Understand ProxyFactoryBean

```
<bean id="loggedTask"
        class="org.springframework.aop.framework.ProxyFactoryBean">
   <property name="proxyInterfaces">
      <list>
         <value>rex.log.LoggedTask</value>
      </list>
   </property>
   <property name="interceptorNames">
      <list>
         <value>loggingAdvice</value>
      </list>
   </property>
   <property name="target" ref="loggedTaskTarget" />
</bean>

<bean id="loggedTaskTarget" class="rex.log.LoggedTask2" />
```

Use Interceptor

```
public class LoggingInterceptor implements MethodInterceptor {

    public Object invoke( MethodInvocation invocation ) throws Throwable
    {
        // do something before method invocation
        …

        Object result = invocation.proceed();

        // do something after method invocation
        …

        return result;
    }
}
```

Configure Static Pointcuts

- NameMatchMethodPointcutAdvisor
- RegExpPointcutAdvisor

AutoProxying

- BeanNameAutoProxyCreator
- DefaultAdvisorAutoProxyCreator

About AOP

- Good??
- Bad??