CS520 Web Programming
Spring – Aspect Oriented Programming

Chengyu Sun
California State University, Los Angeles

Spring Framework

Concerns
User Service  Admin Service  Misc. Service

Cross-Cutting Concerns
User Service  Admin Service  Misc. Service
Logging  Transaction Management  Security

An Example of Transaction Rollback

```java
    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

-_loggedTask1 and LoggedTask2
  - Both implement LoggedTask interface
  - LoggedTask1 - mixes application code and logging code
  - LoggedTask2 - only has application code

How Does it Work?

class Foo {
    Object bar() {...}
}

Proxy – Subclass

class Foo1 extends Foo {
    void bar() {
        // some code before super.bar()
        Object o = super.bar();
        // some code after super.bar();
        return o;
    }
}

Proxy – Wrapper Class

class Foo2 {
    private Foo foo;
    void bar() {
        // some code before super.bar()
        Object o = foo.bar();
        // some code after super.bar();
        return o;
    }
}

Some AOP Terminology

- Target
- Proxy
- Proxy Interface
- Aspect
- Advice

Create Proxies Automatically - ProxyFactoryBean

<bean id="loggedTask2" class="cs520.spring.aop.LoggedTask2" />
<bean id="loggedTask2WithAdvice" class="org.springframework.aop.framework.ProxyFactoryBean">
    <property name="proxyInterfaces">
        <list>
            <value>cs520.spring.log.LoggedTask</value>
        </list>
    </property>
    <property name="target" ref="loggedTask2" />
    <property name="interceptorNames">
        <list>
            <value>loggingAdvice</value>
        </list>
    </property>
</bean>
More AOP Terminology

- Join point: a point in the execution of the application where the aspect can be plugged in
- Pointcut: A predicate that determines join points
- Introduction: adding new methods and/or fields to existing classes
- Weaving
  - Compile time, class load time, or 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Around</td>
<td>org.aopalliance.intercept.MethodInterceptor</td>
</tr>
<tr>
<td>Before</td>
<td>org.springframework.aop.BeforeAdvice</td>
</tr>
<tr>
<td>After</td>
<td>org.springframework.aop.AfterReturningAdvice</td>
</tr>
<tr>
<td>Throws</td>
<td>org.springframework.aop.ThrowsAdvice</td>
</tr>
</tbody>
</table>

Use Interceptor

```java
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 Pointcuts

- NameMatchMethodPointcutAdvisor
- RegExpPointcutAdvisor

AutoProxying

- BeanNameAutoProxyCreator
- DefaultAdvisorAutoProxyCreator
About AOP

- Good??
- Bad??