

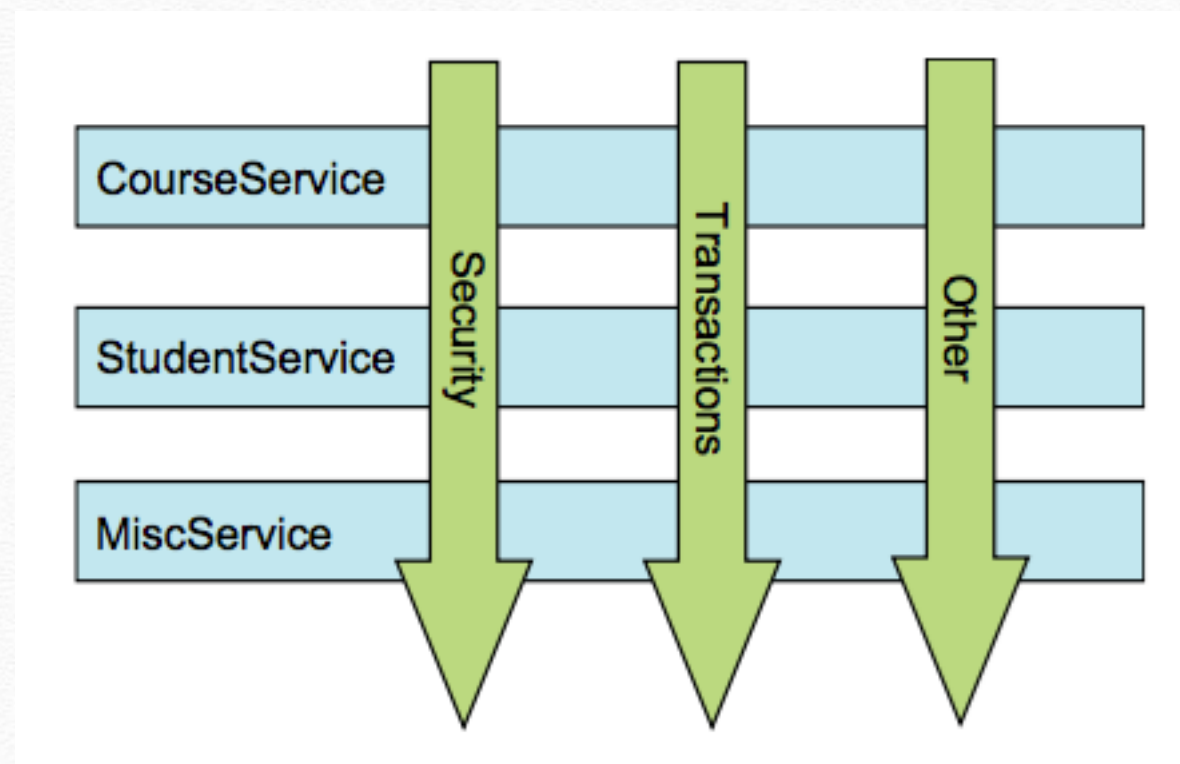


Spring Lesson 6

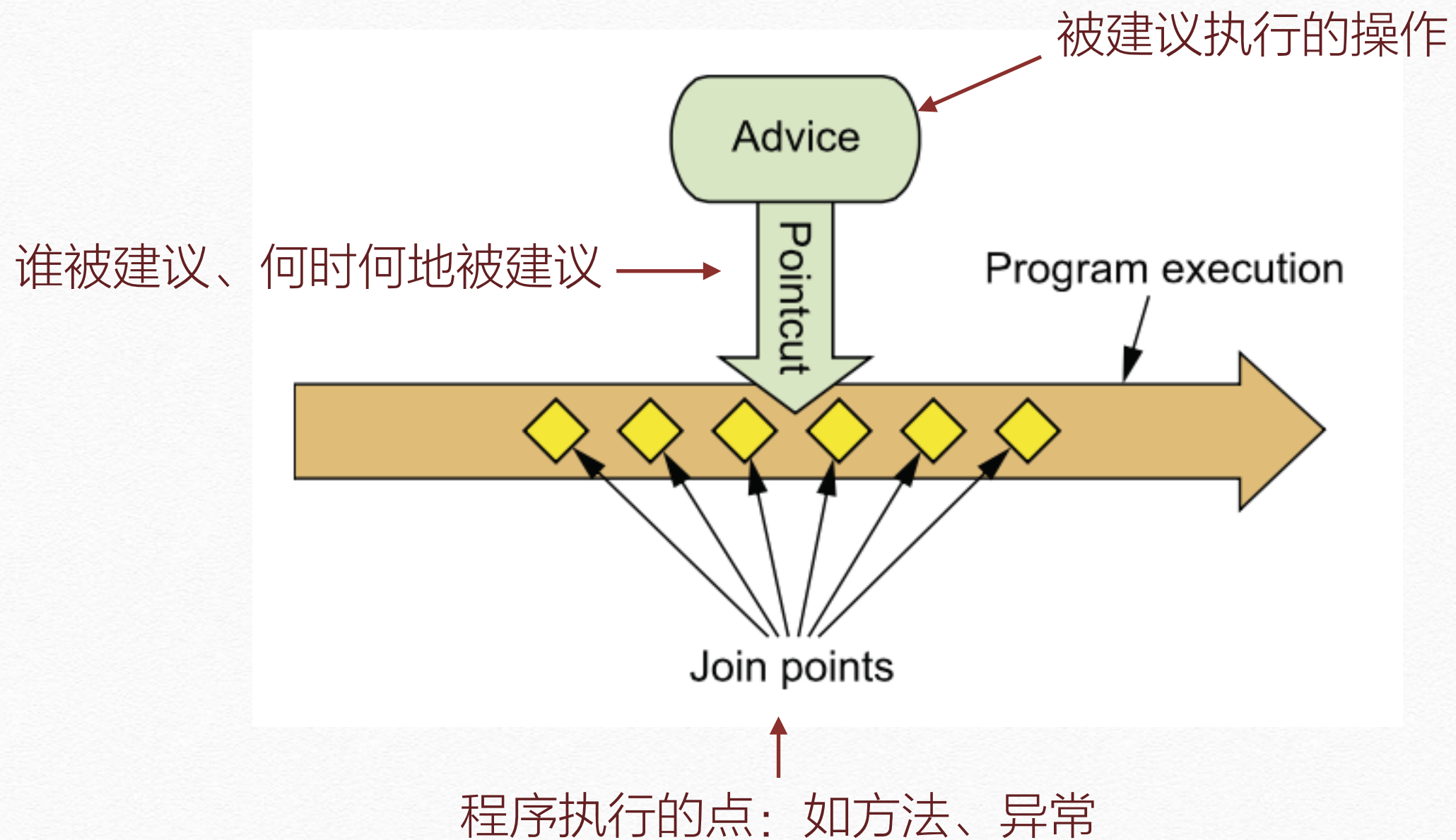
xieqiaoyun

WHY AOP

- ❖ Aspect: 切面(或关注面), 零散而又耦合的公共功能, 分散在各个模块(如 Security、log、Transactions)
- ❖ 传统方式, 改变这些模块(如 Security、log、Transactions)会影响其它所有模块; 面向切面编程(AOP), 即把这些零散的公共方法抽象出来成为一个模块(Asspect), 达到分离关注点, 提升组件可复用性

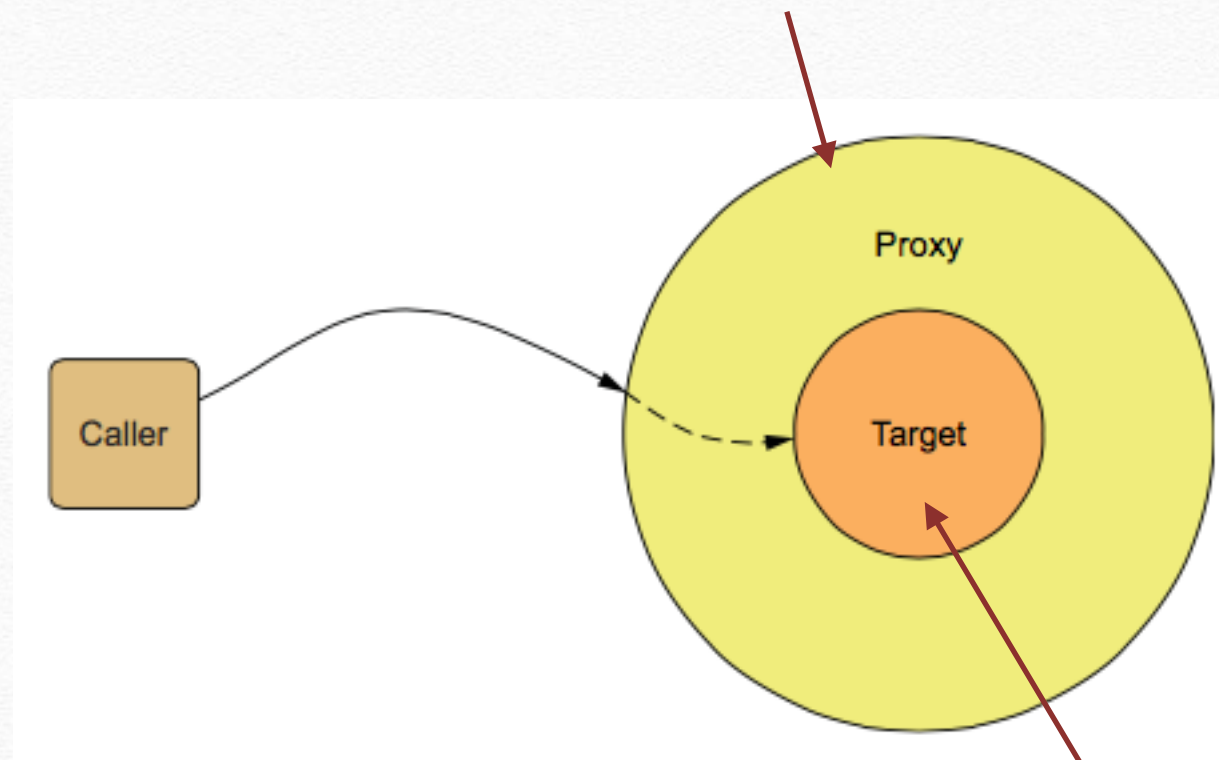


AOP基础概念



AOP基础概念

AOP 代理：遵循Aspect约定的代理对象，可以是JDK代理或CGLIB代理



被拦截(advice)的方法的对象

AOP基础概念

- ❖ Weaving: 编织, 将Pointcut、Advice、AOP proxy和TargetObject等建立连接; 有如下几个阶段实现:
 - ❖ 编译时期
 - ❖ 加载时期(classLoader之前)
 - ❖ 运行时
- ❖ 有两种方式实现:
 - ❖ JDK 动态代理
 - ❖ 修改字节码

原来想象中的AOP

window就是一个容器

```
<script>
(function () {
    function say(msg) {
        console.log(msg);
    }
    function aop_add_log(func) {
        var fn = func.name;
        var _func = window[fn];
        window[fn] = function(){
            console.log("start " + fn);
            _func.apply(window, arguments);
            console.log("end " + fn);
        }
    }

    //test
    aop_add_log(say);
    say("hello");
})();
</script>
```

但是修改原来代码，不符合闭/开原则

asm修改字节码的方式是新生成一个类，继承自原来的类，重新ClassLoader进来

模仿Spring 事务练习AOP

leader.trans目录下的所有以Service结尾的类中：

如果方法名是query开头的，则只自动开启JDBC连接，而不开启事务，如果方法名是save/update/create/delete开头的，则同时开启事务，具体细节如下：

则进入被Advise的方法前，先打开一个JDBC连接，并放入到线程上下文中，方法结束后，则关闭连接，如果有事务，则提交事务，如果业务方法抛出runtime exception，则触发事务的回滚，其他异常则继续提交事务。

模仿Spring 事务练习AOP - (一)

```
@Component
public class MyDataSource {
    DataSource dataSource;
    JdbcTemplate jdbcTemplate;
    PlatformTransactionManager transactionManager;

    public MyDataSource() { this.initDs(); }
    public void initDs(){
        String connectionString = "jdbc:mysql://localhost:3306/leader";
        dataSource = new DriverManagerDataSource(connectionString, username: "root", password: "123456");
        jdbcTemplate = new JdbcTemplate(dataSource);
        transactionManager = new EmployeeTransactionManager(dataSource);
    }
    public Connection getConnection(){
        try {
            System.out.println("—— get connection ...");
            Connection conn = dataSource.getConnection();
        } catch (SQLException e){
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public void closeConnection(){...}
    public DataSource getDataSource() { return dataSource; }
    public JdbcTemplate getJdbcTemplate() { return jdbcTemplate; }
    public PlatformTransactionManager getTransactionManager() { return transactionManager; }
}
```

数据库连接定义类，
数据库连接和事务管理
分开

模仿Spring 事务练习AOP - (一)

```
public class EmployeeTransactionManager extends DataSourceTransactionManager {
    private static final long serialVersionUID = 7938334635313866139L;
    public EmployeeTransactionManager(DataSource dataSource) { super(dataSource); }
    @Override
    public DataSource getDataSource() {
        System.out.println("EmployeeTransactionManager getDataSource called ");
        return super.getDataSource();
    }
    @Override
    protected Object doGetTransaction() {
        System.out.println("EmployeeTransactionManager doGetTransaction called ");
        return super.doGetTransaction();
    }
    @Override
    protected boolean isExistingTransaction(Object transaction) {
        System.out.println("EmployeeTransactionManager isExistingTransaction called ");
        return super.isExistingTransaction(transaction);
    }
    @Override
    protected void doBegin(Object transaction, TransactionDefinition definition) {
        System.out.println("EmployeeTransactionManager doBegin called ");
        super.doBegin(transaction, definition);
    }
    @Override
    protected void doCommit(DefaultTransactionStatus status) {
        System.out.println("EmployeeTransactionManager doCommit called ");
        super.doCommit(status);
    }
    @Override
    protected void doRollback(DefaultTransactionStatus status) {
        System.out.println("doRollback doCommit called ");
        super.doRollback(status);
    }
    @Override
```

程式化事务，自定义
一个
TransactionManager，
继承自
DataSourceTransaction
Manager

模仿Spring 事务练习AOP - (一)

```
@Component
public class EmployeeService {

    @Autowired
    MyDataSource myDataSource;

    public List<Employee> queryEmployees(){
        List<Employee> employees = new ArrayList<>();
        employees = myDataSource.getJdbcTemplate().query( sql: "select * from employee", new Employee
        System.out.println("    ✓ query "+employees.size()+" employees ... ");
        for(Employee employee:employees){
            System.out.println("        "+employee);
        }
        return employees;
    }

    public void saveEmployee(Employee e){
        System.out.println("    try to update employee ... ");
        Employee employee = myDataSource.getJdbcTemplate().queryForObject( sql: "select * from employ
            new Object[]{new Integer(e.getEmpid())},new EmployeeMapper());
        System.out.println("        "+employee);
        if(employee.getEmpid() == 0){
            throw new RuntimeException("A Runtime Exception...");
        }
        System.out.println("    ✓ save employee ... ");
    }

    public void updateEmployee(int empid, long salary){
        System.out.println("    try to update employee ... "+empid);
        Employee employee = myDataSource.getJdbcTemplate().queryForObject( sql: "select * from employ
            new Object[]{new Integer(empid)},new EmployeeMapper());
        System.out.println("        "+employee);
        if(employee.getEmpid() == 0){
            throw new RuntimeException("A Runtime Exception...");
        }
    }
}
```

Service主要是
是数据库增
删改查操作,

JdbcTemplate
坑了好长时
间...

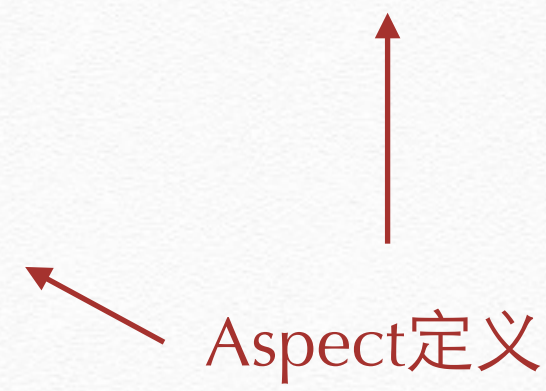
模仿Spring 事务练习AOP - (一)

```
Aspect
@Component
public class EmployeeJdbcAspect {
    @Autowired
    MyDataSource myDataSource;

    @Around("execution(* leader.trans.*Service.get*(..)) " +
            "|| execution(* leader.trans.*Service.save*(..)) " +
            "|| execution(* leader.trans.*Service.update*(..)) " +
            "|| execution(* leader.trans.*Service.create*(..)) " +
            "|| execution(* leader.trans.*Service.delete*(..)) ")
    public Object employeeAroundAdvice(ProceedingJoinPoint proceedingJ
        System.out.println("—— Around Advice: Before invoking method ——");
        Object value = null;
        Object target = null;
        // 数据库连接
        myDataSource.getConnection();
        // 开启事务
        DefaultTransactionDefinition transactionDefinition = new DefaultTransact
        transactionDefinition.setPropagationBehavior(TransactionDefinition.PROPA
        TransactionStatus status = myDataSource.getTransactionManager().getTrans
        try {
            target = proceedingJoinPoint.getTarget();
            value = proceedingJoinPoint.proceed();
            // 提交事务
            myDataSource.getTransactionManager().commit(status);
        } catch (Throwable e) {
            if(e instanceof RuntimeException){
                System.out.println("—— Found RuntimeException in "+target.getCl
                // 回滚
                myDataSource.getTransactionManager().rollback(status);
            } else {
                e.printStackTrace();
                // 关闭连接
                myDataSource.closeConnection();
            }
        }
    }

    @Before("execution(* leader.trans.*Service.query*(..))")
    public void employeeBeforeAdvice(){
        System.out.println("—— Accept Before Advice ...");
        myDataSource.getConnection();
    }

    @After("execution(* leader.trans.*Service.query*(..))")
    public void employeeAfterAdvice(){
        System.out.println("—— Accept After Advice ...");
        System.out.println();
    }
}
```



Aspect定义

模仿Spring 事务练习AOP - (一)

```
@Configuration
@ComponentScan("leader.trans")
@EnableAspectJAutoProxy ← 启用Spring AOP 代理
public class App {

    public static void main(String[] args) {
        ApplicationContext ctx = new AnnotationConfigApplicationContext(App.class);
        EmployeeService employeeService = ctx.getBean(EmployeeService.class);

        employeeService.getEmployee( empid: 1);
        employeeService.queryEmployees();

        int empid = (int) (Math.random()*100);
        employeeService.createEmployee(new Employee(empid, name: "employee"+empid));

        employeeService.updateEmployee(empid, (long) (Math.random()*10000));

        employeeService.saveEmployee(new Employee( empid: empid+1, name: "employee"+empid+1, (int)

//        employeeService.deleteEmployee(empid+1);
        System.out.println("—— 增删改查之后, rollback之前的数据 ——");
        employeeService.queryEmployees();

        employeeService.deleteEmployee( empid: 0);

        System.out.println("—— 增删改查之后, rollback之后的数据 ——");
        employeeService.queryEmployees();
    }
}
```

查询, 执行操作, 回滚, 查询数据

预先插入几条数据

```
//create table employee(empid int(4),name varchar(50),age int(2),salary int(11),primary key(empid));
//insert into employee(empid,name,age,salary) values(1,"emp001",22,2200);
//insert into employee(empid,name,age,salary) values(2,"emp002",23,2200);
//insert into employee(empid,name,age,salary) values(3,"emp003",25,3200);
```


模仿Spring 事务练习AOP - 运行结果

```
EmployeeTransactionManager getDataSource called
--- Around Advice: Before invoking method ---
--- get connection ...
Thu Sep 21 18:34:22 CST 2017 WARN: Establishing SSL connection
EmployeeTransactionManager doGetTransaction called
EmployeeTransactionManager isExistingTransaction called
EmployeeTransactionManager doBegin called
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection
EmployeeTransactionManager getDataSource called
- try to get employee ...
EMPLOYEE{empid- 1 name- emp001 age- 22 salary- 2200}
√ get employee ...
EmployeeTransactionManager doCommit called
--- Around Advice: After invoking method. Return value=EMPLOYE
--- Accept Before Advice ...
--- get connection ...
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection
√ query 3 employees ...
EMPLOYEE{empid- 1 name- emp001 age- 22 salary- 2200}
EMPLOYEE{empid- 2 name- emp002 age- 23 salary- 2200}
EMPLOYEE{empid- 3 name- emp003 age- 25 salary- 3200}
--- Accept After Advice ...

--- Around Advice: Before invoking method ---
--- get connection ...
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection
EmployeeTransactionManager doGetTransaction called
EmployeeTransactionManager isExistingTransaction called
EmployeeTransactionManager doBegin called
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection
EmployeeTransactionManager getDataSource called
try to create employee ...
√ create employee ...
EmployeeTransactionManager doCommit called
--- Around Advice: After invoking method. Return value=null ---
```

```
--- 增删改查之后, rollback之前的数据 ---
--- Accept Before Advice ...
--- get connection ...
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection w
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection w
√ query 4 employees ...
EMPLOYEE{empid- 1 name- emp001 age- 22 salary- 2200}
EMPLOYEE{empid- 2 name- emp002 age- 23 salary- 2200}
EMPLOYEE{empid- 3 name- emp003 age- 25 salary- 3200}
EMPLOYEE{empid- 35 name- employee35 age- 0 salary- 1916}
--- Accept After Advice ...

--- Around Advice: Before invoking method ---
--- get connection ...
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection w
EmployeeTransactionManager doGetTransaction called
EmployeeTransactionManager isExistingTransaction called
EmployeeTransactionManager doBegin called
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection w
EmployeeTransactionManager getDataSource called
try to delete employee ...
--- Found RuntimeException in aoptest.EmployeeService$$EnhancerB
doRollback doCommit called
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection w
--- Around Advice: After invoking method. Return value=null ---

--- 增删改查之后, rollback之后的数据 ---
--- Accept Before Advice ...
--- get connection ...
Thu Sep 21 18:34:23 CST 2017 WARN: Establishing SSL connection w
√ query 4 employees ...
EMPLOYEE{empid- 1 name- emp001 age- 22 salary- 2200}
EMPLOYEE{empid- 2 name- emp002 age- 23 salary- 2200}
EMPLOYEE{empid- 3 name- emp003 age- 25 salary- 3200}
EMPLOYEE{empid- 35 name- employee35 age- 0 salary- 1916}
--- Accept After Advice ...
```


模仿Spring 事务练习AOP - (二)

本来还想用asm修改字节码的...

Thank You

xieqiaoyun