## 注解：Annotation

Java 5.0 新增注解特性

注解是代码的附属信息，遵循一个基本原则：注解不能直接干扰程序代码的运行，无论增加或删除注解，代码都能够正常运行。

Java语言解释器会忽略这些注解，而由第三方工具负责对注解进行处理。

@interface修饰符定义注解，一个注解可以拥有多个成员，成员声明和接口方法声明类似，成员声明有以下几点限制：

1.成员以无入参，无抛出异常的方式声明

2.可以通过default为成员指定默认值

3.成员类型只能是基本数据类型及其包装类型、String、Class、enum、注解类型

4.如果只有一个成员，则成员的名字必须取为value()

元注解(Meta-Annotation)：Java预定义的注解，被Java编译器使用，会对注解类的行为产生影响。常用的元注解有：

@Retention(RetentionPolicy.RUNTIME) – 注解类信息保留期限

1. RetentionPolicy.SOURCE:注解信息仅保存在目标类的源码文件中，对应的字节码文件将不再保留

2. RetentionPolicy.CLASS:注解信息将进入目标类的字节码文件中，但类加载器加载字节码文件时不会将注解加载到JVM中，即运行期不能获取注解信息

3. RetentionPolicy.RUNTIME:注解信息在目标类加载到JVM后依然保留，在运行期可以通过反射机制读取类中的注解信息。

@Target(ElementType.METHOD):注解类的应用范围

## 反射：Reflect

## 数据库事务基础知识

**事务的特性**

数据库事务的4个特性：原子性(Atomic)、一致性(Consistency)、隔离性(Isolation)、持久性(Durability),简称ACID

原子性：组成事务的多个数据库操作是一个不可分割的原子单元；

一致性：事务操作成功后，数据库所处的状态和它的业务规则是一致的；

隔离性：并发数据操作时，不同的事务拥有各自的数据空间，操作不会对对方产生干扰；

持久性：一旦事务提交成功后，事务中的所有数据操作都必须被持久化到数据库中。

**数据并发的问题**

脏读(dirty read)、不可重复读、幻象读、第一类丢失更新、第二类丢失更新。

1.脏读：

## ThreadLocal

如果某个对象是非线程安全的，在多线程环境下，对对象的访问必须采用synchronized进行线程同步；线程同步会降低并发性，影响系统性能。

TreadLocal为解决多线程程序的并发问题提供一种新的思路；ThreadLocal是一个保存线程本地化对象的容器；当运行于多线程环境的某个对象使用ThreadLocal维护变量时，ThreadLocal为每个使用该变量的线程分配一个独立的变量副本。

ThreadLocal和线程同步机制都是为了解决多线程中相同变量的访问冲突问题。

在同步机制中，通过对象的锁机制保证同一时间只有一个线程访问变量，该变量是多个线程共享的；

ThreadLocal为每个线程提供一个独立的变量副本，从而隔离了多个线程对访问数据的冲突；

概况而言，对于多线程资源共享的问题，同步机制采用“时间换空间”的方式：访问串行化，对象共享化；而ThreadLocal采用“空间换时间”的方式：访问并行化，对象独立化。