# 课程回顾

Xml技术

Servlet/jsp

数据库

Jdbc技术

javaWeb增强

过滤器，监听器，文件上传与下载，国际化

学过的组件：

Dom4j/xmpath 用来解析xml的jar包

Dbutils 用来简化对数据库操作

C3p0 连接池技术

BeanUtils javaBean相关，可以把map集合转化成bean

今天的目标：

1. 泛型
2. 注解
3. Log4j 日志jar包

# 泛型

掌握要求：

基本用法、泛型擦除、泛型类/泛型方法/泛型接口、反射泛型（案例）、泛型关键字

## 概述

泛型是jdk1.5以后才有的，可以在编译时期进行类型检查，且可以避免频繁转换类型。

|  |
| --- |
| public void test(){  List<String > list=new ArrayList<String>();  list.add("abc");    // list.add(1); //编译时期报错    String str=list.get(0);  } |

泛型擦除

泛型只在编译时期有效，（即写代码时有效），而在编译后的字节码文件中是不存在的。

|  |
| --- |
| /\*  \* 泛型擦除实例  \* \*/  public void save (List<Person p>){    }    public void save(List<Student s>){ //报错，与上面方法编译后一样，泛型在编译后是不存在的，编译后，传递的参数都是List ，所以报错    } |

泛型不能为基本类型。

## 泛型方法/泛型类/泛型接口

作用：

设计公用的类、方法，对公用的业务实现进行抽取，使程序更灵活。

泛型方法

|  |
| --- |
| public <T,K> T save(T dept ,K k){ //这样就是定义泛型。  return null;  }    //测试方法  @Test  public void test() throws Exception{  save(1.0f,1); //泛型类型确定的时间：什么时候使用泛型方法，什么时候可以确定泛型类型。  } |

泛型类/泛型接口

|  |
| --- |
| //泛型类：在创建泛型对象的时候，确定类型。或者被继承时。  GernericDmeo<String> g=new GernericDmeo<String>();  g.save("", 1);  }  // 泛型接口： 实现泛型接口的类也是抽象，那么类型在具体的实现类中确定或创建泛型类的时候确定。 |

## c.泛型的关键字

泛型中 ？ extends super

关键字： ？

|  |
| --- |
| public void save(List<?> list){  //只能获取，迭代list，不能编辑list  }    public void testGeneric() throws Exception{  //? 可以接受任何泛型集合，但是不能编辑集合值，所以一般在方法参数中使用。    List<?> list=new ArrayList<String>();  } |

关键字 extends

|  |
| --- |
| /\*  \* list集合只能处理Double，float，intefer等类型  \* 限定元素范围：元素的类型要继承自Numer类 (上限)  \* \*/  public void save(List<? extends Number> list){  //这样写  }    public void testGeneric() throws Exception{  //? 可以接受任何泛型集合，但是不能编辑集合值，所以一般在方法参数中使用。    List<Double> list=new ArrayList<Double>();  List<Float> list1=new ArrayList<Float>();  List<Integer> list2=new ArrayList<Integer>();      List<String> list3=new ArrayList<String>();    save(list);  // save(list3); 这个就不行，因为list3 集合中元素属于String 不是继承自 Number  } |

\*由于定义多态的时候，多是定义父类，那么extends 关键字限定只能传自己以及自己的子类的用法就很多。 而像super这样的用的就很少。

## d．泛型的反射

案例，设置通用方法，会用到反射泛型！。

步骤：

1. 案例分析/ 实现
2. 涉及知识点
3. 优化 /用反射泛型

案例：

要从两个不同的表中查询数据，主键相同，那么通过主键查询数据，是否可以有一个通用功能呢？

反射泛型涉及的api：

Type： Type是java编程语言中所有类型的公共高级接口。包括：基本类型，引用类型，

**参数化类型**

**ParameterizedType 参数化类型的表示**

List<String> list=new ArrayList<String>();

泛型集合： list

集合元素定义： String

参数化类型：ArrayList<String> ParameterizedType.

什么是反射？

Class clazzz=Class.forName(className); //通过名字获得类的字节码文件。

根据字节码文件获得对象

Admin admin=（Admin）clazz.newInstance(); //默认构造函数。

Constructor constructor=clazz.getConstructor(String.class,int.class)



# 3.注解

## 概述

注解与注释，

注解，告诉编译器如何运行程序。

注释，告诉程序员阅读的，对运行什么的没有影响。

## 自定义注解

写法：

|  |
| --- |
| public @interface Author {  /\*  \* 注解属性  \* 1.修饰为默认或者public  \* 2.方法不能有主体｛｝  \* \*/  String name();  int age();  } |

使用：

|  |
| --- |
| @Author(name ="zk" , age =23 )  public class AnnotationDemo {  public void save(){  }  } |

## 元注解

注解的注解，修饰注解的注解。

@Target({TYPE, METHOD}) 用来定义注解的范围

@Retention 定义注解的生命周期。

Retention.SOURCE 注解只在源码级别有效

Retention.CLASS 注解只在字节码有效

Retention.RUNTIME 注解在运行时期有效 （范围最大）

## 获取注解信息(注解反射)

先获取代表方法的Method对象，然后通过

Author author=method.getAnnotation(Author.class); 获取到注解Author对象。

# Log4j日志jar使用

程序中为什么用日志组件

简单来说，为了项目后期部署上线后的维护、错误排查。

Log4j.JAR 开源日志组件。

使用步骤：

1. 导包

导入log4j-1.2.11.jar

1. 配置

|  |
| --- |
| **log4j.rootLogger** =debug,console, file  #2. 日志输出到控制台使用的api类。 **log4j.appender.console**=org.apache.log4j.ConsoleAppender  #日志输出格式 **log4j.appender.console.layout**=org.apache.log4j.PatternLayout #---具体格式内容 **log4j.appender.console.layout.ConversionPattern**=%d %p %c.%M()-%m%n  #--------日志输出到文件------------------------------- #---日志输出到文件采用的api， 作用是 文件大小到达一定尺寸的时候产生一个新的文件。 **log4j.appender.file**=org.apache.log4j.RollingFileAppender #指定日志文件路径 **log4j.appender.file.File**=../logs/myLog.log #指定日志文件最大大小 **log4j.appender.file.MaxFileSize**=1024KB #指定日志文件最大数目 **log4j.appender.file.MaxBackupIndex**=10 **log4j.appender.file.layout**=org.apache.log4j.PatternLayout **log4j.appender.file.layout.ConversionPattern**=%d %p %c.%M()-%m%n |

# 动态代理

1. 代理模式

代理模式的作用：

屏蔽真实行为的访问，让程序更加安全。

可以对真实行为的调用进行控制。