异常、日志技术、阶段项目

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

今天同学们需要学会什么

程序在编译阶段、或者运行时可能出现错误。

系统在开发测试阶段或者上线后可能会出bug？这些bug是不是需要记录下来，方便根据记录信息修正bug。

之前学了很多面向对象语法，还学习了很多集合，这些东西在项目开发中用的很多，很重要。

目录

Contents

- 异常
  - 异常概述、体系
  - 常见运行时异常
  - 常见编译时异常
  - 异常的默认处理流程
  - 编译时异常的处理机制
  - 运行时异常的处理机制
  - 异常处理使代码更稳健的案例
  - 自定义异常
- 日志技术
- 阶段项目

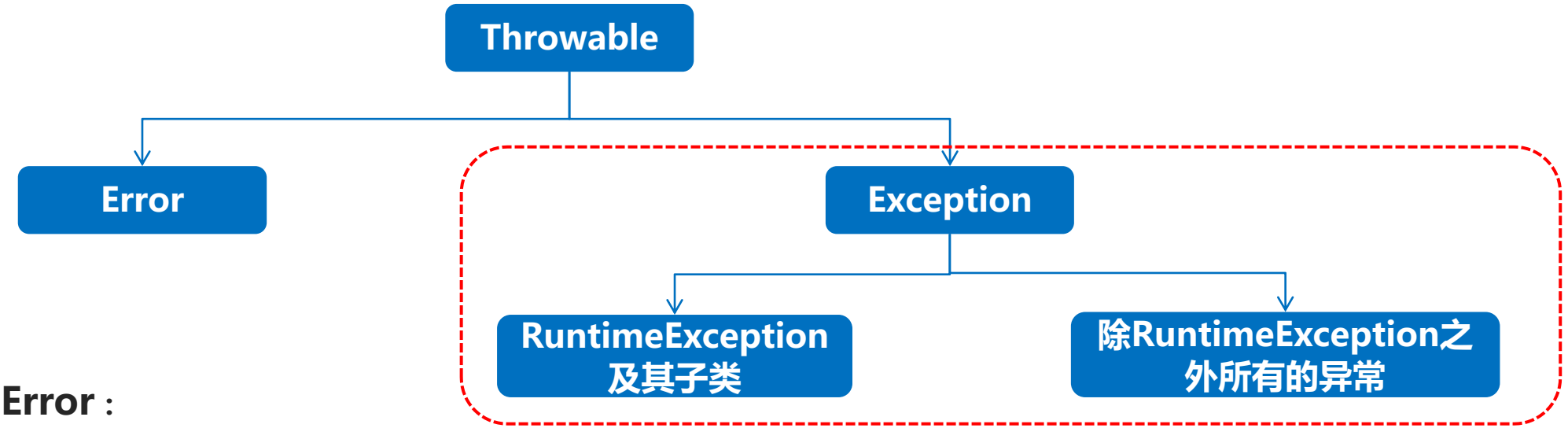
什么是异常？

- 异常是程序在“编译”或者“执行”的过程中可能出现的问题，**注意：**语法错误不算在异常体系中。
- 比如:数组索引越界、空指针异常、 日期格式化异常，等…

为什么要学习异常?

- 异常一旦出现了，如果没有提前处理，程序就会退出JVM虚拟机而终止.
- 研究异常并且避免异常，然后提前处理异常，体现的是程序的安全，健壮性。

异常体系



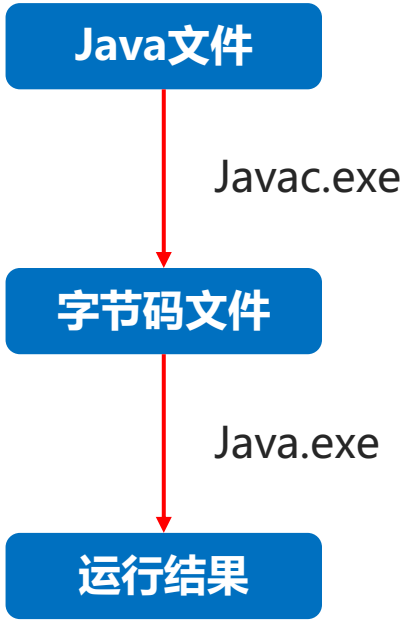
Error：

- 系统级别问题、JVM退出等，代码无法控制。

Exception：java.lang包下，称为异常类，它表示程序本身可以处理的问题

- **RuntimeException及其子类**：运行时异常，编译阶段不会报错。(空指针异常，数组索引越界异常)
- **除RuntimeException之外所有的异常**：编译时异常，编译期必须处理的，否则程序不能通过编译。(日期格式化异常)。

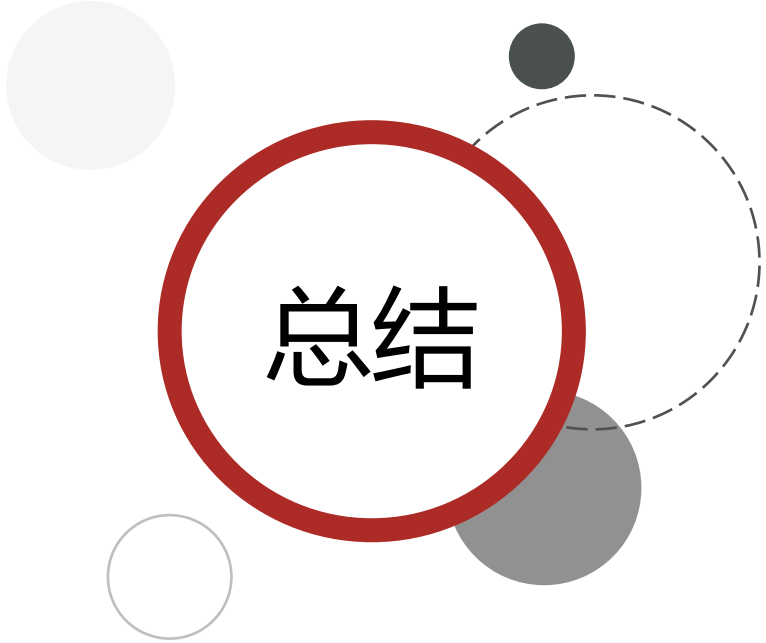
编译时异常和运行时异常



- 编译时异常，是在编译成class文件时必须要处理的异常，也称之为受检异常
- 运行时异常，在编译成class文件不需要处理，在运行字节码文件时可能出现的异常。

简单来说：

- 编译时异常就是在编译的时候出现的异常，
- 运行时异常就是在运行时出现的异常。



1. 异常是什么？
   - **异常是代码在编译或者执行的过程中可能出现的错误。**
2. 异常分为几类？
   - **编译时异常、运行时异常。**
   - **编译时异常：没有继承RuntimeExcpetion的异常，编译阶段就会出错。**
   - **运行时异常：继承自RuntimeException的异常或其子类，编译阶段不报错，运行可能报错。**
3. 学习异常的目的？
   - **避免异常的出现，同时处理可能出现的异常，让代码更稳健。**

目录

Contents

运行时异常

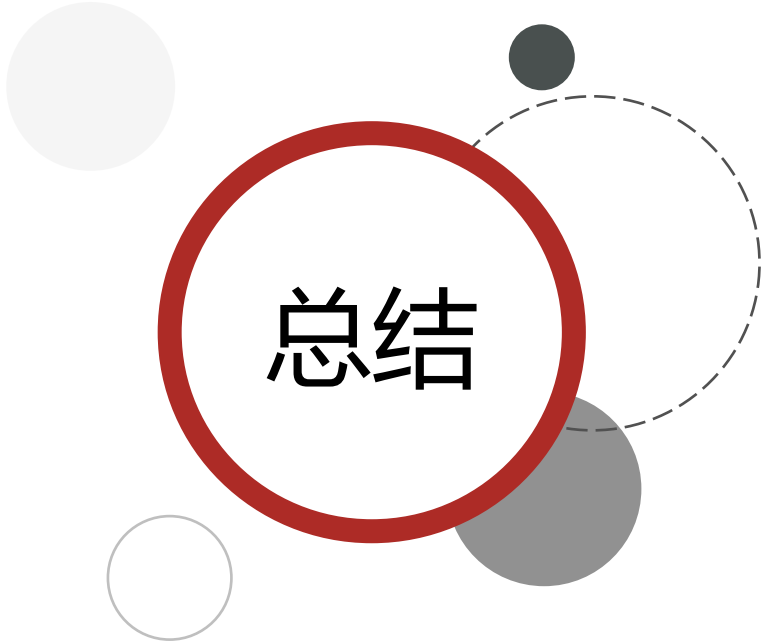
- **直接继承自RuntimeException或者其子类，**编译阶段不会报错，运行时可能出现的错误。

运行时异常示例

- 数组索引越界异常: ArrayIndexOutOfBoundsException
- 空指针异常：NullPointerException，直接输出没有问题，但是调用空指针的变量的功能就会报错。
- 数学操作异常：ArithmeticException
- 类型转换异常：ClassCastException
- 数字转换异常： NumberFormatException

运行时异常：一般是程序员业务没有考虑好或者是编程逻辑不严谨引起的程序错误，

自己的水平有问题！



1. 运行时异常的特点

- **运行时异常：继承自RuntimeException的异常或者其子类，**
- **编译阶段不报错,运行可能报错。**

目录

Contents

编译时异常

- **不是RuntimeException或者其子类的异常，**编译阶就报错，必须处理，否则代码不通过。

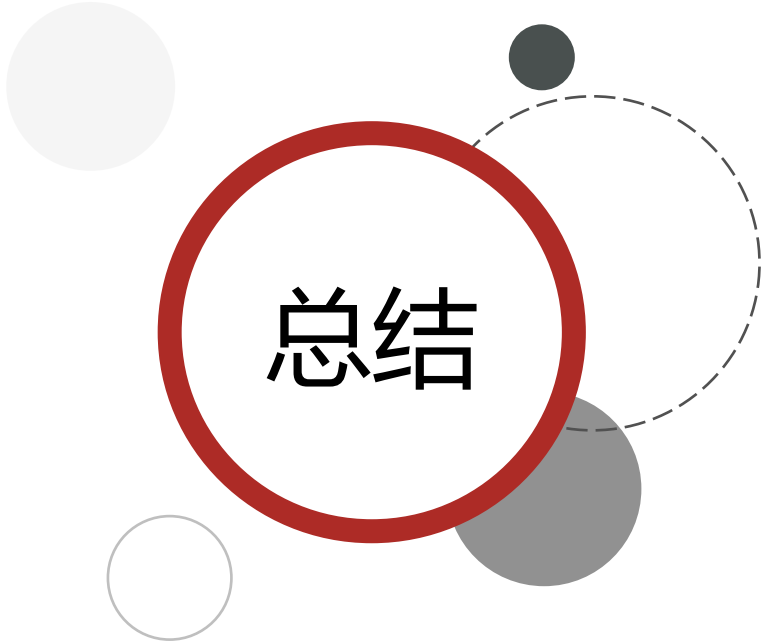
编译时异常示例

```
String date = "2015-01-12 10:23:21";
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
Date d = sdf.parse(date);
System.out.println(d);
```

日期解析异常：ParseException

编译时异常的作用是什么：

- 是担心程序员的技术不行，在编译阶段就爆出一个错误, 目的在于提醒不要出错!
- 编译时异常是可遇不可求。遇到了就遇到了呗。



1. 编译时异常的特点

- **编译时异常：继承自Exception的异常或者其子类**
- **编译阶段报错，必须处理，否则代码不通过。**

目录

Contents

① 默认会在出现异常的代码那里自动的创建一个异常对象：ArithmeticException。

② 异常会从方法中出现的点这里抛出给调用者，调用者最终抛出给JVM虚拟机。

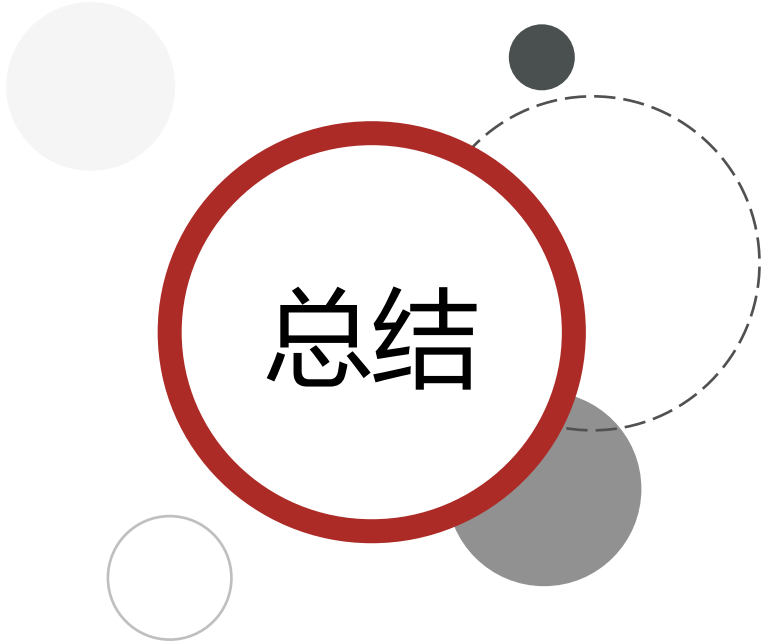
③ 虚拟机接收到异常对象后，先在控制台直接输出异常栈信息数据。

④ 直接从当前执行的异常点干掉当前程序。

⑤ 后续代码没有机会执行了，因为程序已经死亡。

```
public class ExceptionDemo {
    public static void main(String[] args) {
        System.out.println("程序开始。。。。。。。。。");
        chu( a: 10 , b: 0);
        System.out.println("程序结束。。。。。。。。。");
    }

    public static void chu(int a , int b){
        System.out.println(a);
        System.out.println(b);
        int c = a / b ;// 出现了运行时异常,自动创建异常对象：ArithmeticException
        System.out.println("结果是："+c);
    }
}
```



1. 默认异常处理机制。
   - **默认的异常处理机制并不好，一旦真的出现异常，程序立即死亡！**

目录

Contents

- 异常
  - 异常概述、体系
  - 常见运行时异常
  - 常见编译时异常
  - 异常的默认处理流程
  - 编译时异常的处理机制
  - 运行时异常的处理机制
  - 异常处理使代码更稳健的案例
  - 自定义异常
- 日志技术
- 阶段项目

编译时异常是编译阶段就出错的，所以必须处理，否则代码根本无法通过

编译时异常的处理形式有三种：

- 出现异常直接抛出去给调用者，调用者也继续抛出去。
- 出现异常自己捕获处理，不麻烦别人。
- 前两者结合，出现异常直接抛出去给调用者，调用者捕获处理。

异常处理方式1 —— throws

- throws：用在方法上，可以将方法内部出现的异常抛出去给本方法的调用者处理。
- 这种方式并不好，发生异常的方法自己不处理异常，如果异常最终抛出去给虚拟机将引起程序死亡。

抛出异常格式：

```
方法 throws 异常1 ，异常2 ，异常3 ..{
}
```

规范做法：

```
方法 throws Exception{
}
```

- 代表可以抛出一切异常，

异常处理方式2 —— try...catch...

- 监视捕获异常，用在方法内部，可以将方法内部出现的异常直接捕获处理。
- 这种方式还可以，发生异常的方法自己独立完成异常的处理，程序可以继续往下执行。

格式：

```
try{
    // 监视可能出现异常的代码！
}catch(异常类型1 变量){
    // 处理异常
}catch(异常类型2 变量){
    // 处理异常
}...
```

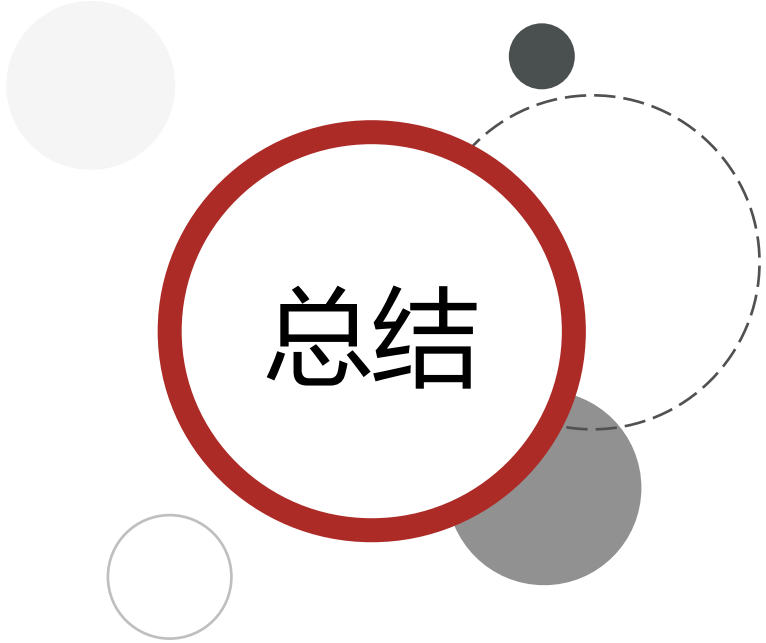
建议格式：

```
try{
    // 可能出现异常的代码！
}catch (Exception e){
    e.printStackTrace(); // 直接打印异常栈信息
}

Exception可以捕获处理一切异常类型！
```

异常处理方式3 —— 前两者结合

- 方法直接将异通过throws抛出去给调用者
- 调用者收到异常后直接捕获处理。



1、异常处理的总结

- **在开发中按照规范来说第三种方式是最好的：底层的异常抛出去给最外层，最外层集中捕获处理。**

- **实际应用中，只要代码能够编译通过，并且功能能完成，那么每一种异常处理方式似乎也都是可以的。**

目录

Contents

运行时异常的处理形式

- 运行时异常编译阶段不会出错，是运行时才可能出错的，所以编译阶段不处理也可以。
- 按照规范建议还是处理：建议在最外层调用处集中捕获处理即可。

目录

Contents

案例 异常处理使代码更稳健的案例

需求

- 键盘录入一个合理的价格为止（必须是数值）。

分析

- 定义一个死循环，让用户不断的输入价格。

目录

Contents

自定义异常的必要？

- Java无法为这个世界上全部的问题提供异常类。
- 如果企业想通过异常的方式来管理自己的某个业务问题，就需要自定义异常类了。

自定义异常的好处：

- 可以使用异常的机制管理业务问题，如提醒程序员注意。
- 同时一旦出现bug，可以用异常的形式清晰的指出出错的地方。

自定义异常的分类

1、自定义编译时异常

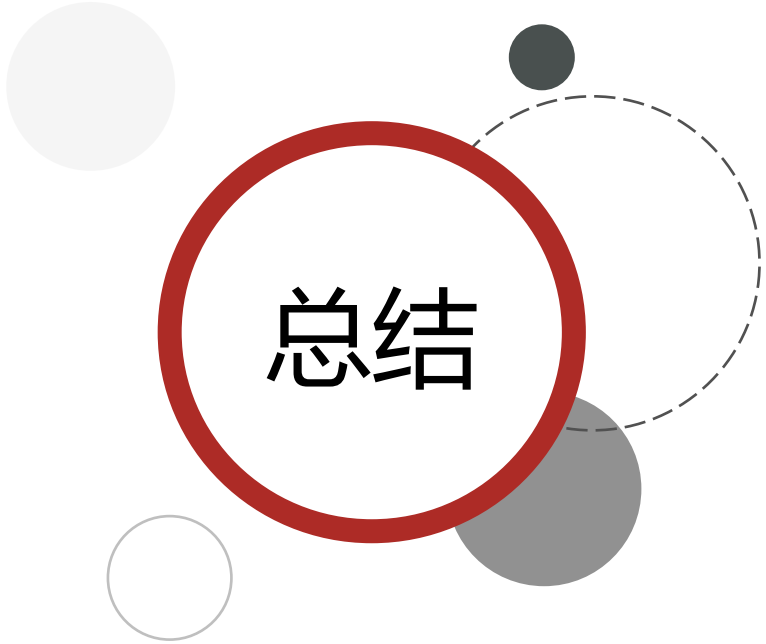
- 定义一个异常类继承Exception.
- 重写构造器。
- 在出现异常的地方用throw new 自定义对象抛出，

作用：编译时异常是编译阶段就报错，提醒更加强烈，一定需要处理！！

2、自定义运行时异常

- 定义一个异常类继承RuntimeException.
- 重写构造器。
- 在出现异常的地方用throw new 自定义对象抛出!

作用：提醒不强烈，编译阶段不报错！！运行时才可能出现！！



1、自定义编译时异常

- 定义一个异常类继承Exception.
- 重写构造器。
- 在出现异常的地方用throw new 自定义对象抛出，

作用：编译时异常是编译阶段就报错，提醒更加强烈，一定需要处理！！

2、自定义运行时异常

- 定义一个异常类继承RuntimeException.
- 重写构造器。
- 在出现异常的地方用throw new 自定义对象抛出!

作用：提醒不强烈，编译阶段不报错！！运行时才可能出现！！

目录

Contents

- 希望系统能记住某些数据是被谁操作的，比如被谁删除了？
- 想分析用户浏览系统的具体情况，以便挖掘用户的具体喜好？
- 当系统在开发或者上线后出现了bug，崩溃了，该通过什么去分析、定位bug？

日志

什么是日志？

- 用来记录程序运行过程中的信息，并可以进行永久存储。好比生活中的日记，可以记录你生活的点点滴滴。

目前记录日志的方案

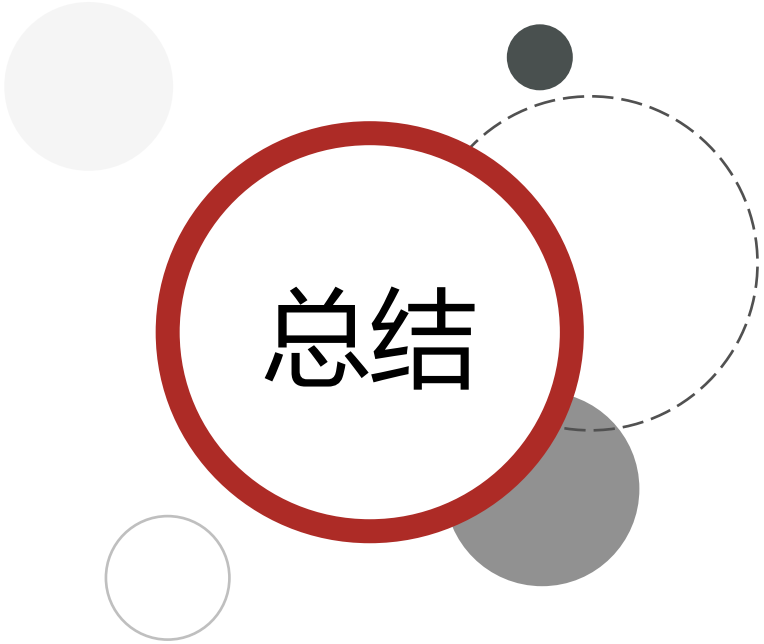
```
Scanner sc = new Scanner(System.in);
System.out.println("请输入一个整数");
String number = sc.nextLine();
try {
    int result = Integer.parseInt(number);
    System.out.println("输入的数字为" + result);
} catch (NumberFormatException e) {
    System.out.println("输入的数字有误，请输入一个整数");
}
```

输出语句的弊端

- 信息展示在控制台
- 不能方便的将其记录到其他的位置（文件，数据库）
- 想取消记录的信息需要修改代码才可以完成

日志技术应该具备哪些特点和优势

- 可以将系统执行的信息，方便的记录到指定的位置（控制台、文件中、数据库中）。
- 可以随时以开关的形式控制是日志的记录和取消，无需侵入到源代码中去进行修改。



1. 什么是日志？

- **用来记录程序运行过程中的信息，并可以进行永久存储。**

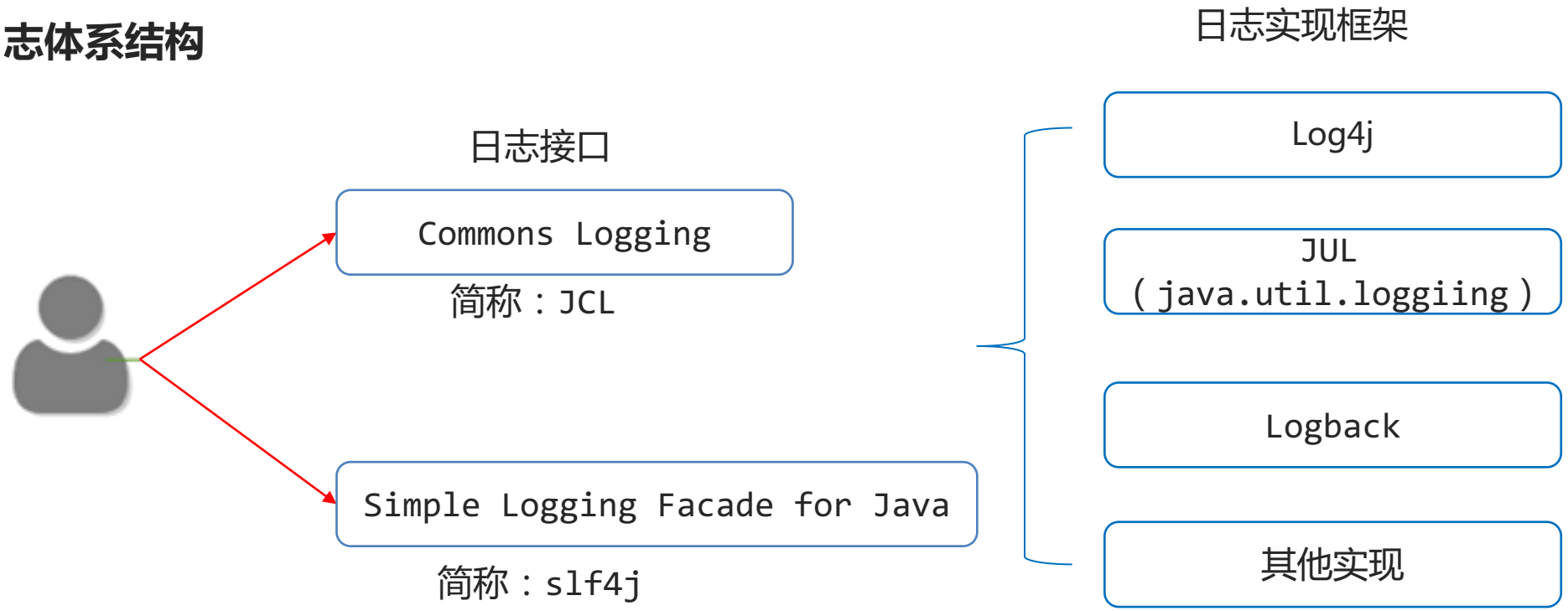
2. 输出语句存在的问题，日志技术应该具备哪些特点和优势？

	输出语句	日志技术
输出位置	输出到控制台	可以将日志信息写入到文件或者数据库中
取消日志	需要修改代码，灵活性比较差	不需要修改代码，灵活性比较好
多线程	性能较差	性能较好

目录

Contents

日志体系结构



- **日志接口：**一些规范，提供给日志的实现框架设计的标准。
- **日志框架：**牛人或者第三方公司已经做好的实现代码，后来者直接可以拿去使用。
- 注意：因为对Commons Logging接口不满意，有人就搞了SLF4J。因为对Log4j的性能不满意，有人就搞了Logback，**Logback是基于slf4j的日志规范实现的框架。**

Logback日志框架

- 官方网站：https://logback.qos.ch/index.html

Logback日志框架分为以下模块：

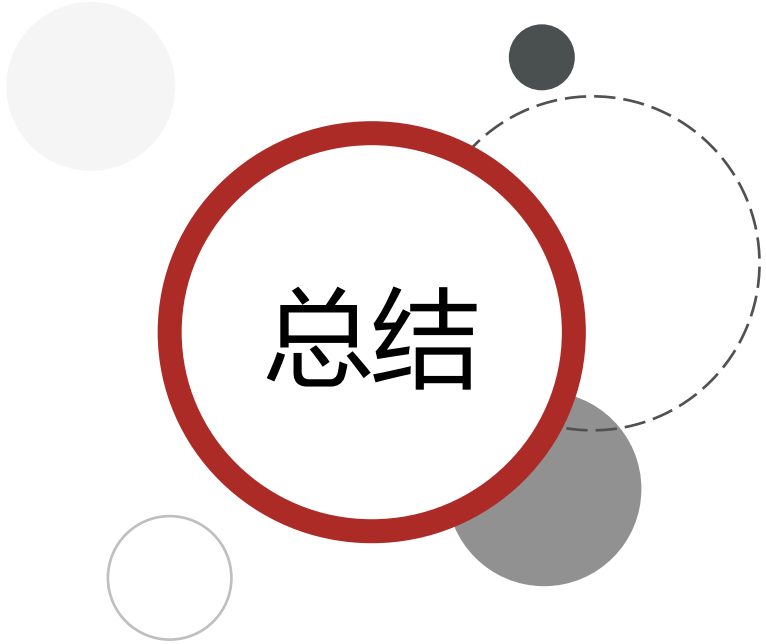
- logback-core： 该模块为其他两个模块提供基础代码。（必须有）
- logback-classic：完整实现了slf4j API的模块。（必须有）
- logback-access 模块与 Tomcat 和 Jetty 等 Servlet 容器集成，以提供 HTTP 访问日志功能（可选模块，以后接触）

想使用Logback日志框架，至少需要在项目中整合如下三个模块：

slf4j-api：日志接口

logback-core：基础模块

logback-classic：功能模块，它完整实现了slf4j API



1. 日志接口是什么，常见的有几种形式。
   - **日志接口大多是一些规范，用来约束日志实现框架的设计。**
   - **Commons Logging、Simple Logging Facade for Java (slf4j)**
2. 常见的日志实现框架有哪些？
   - **Log4J、Logback(我们重点学习的，其他的都大同小异)。**
   - **Logback是基于slf4j日志接口实现的日志框架。**
3. 使用Logback至少需要使用哪几个模块，各自的作用是什么？
   - **slf4j-api：日志接口**
   - **logback-core：基础模块**
   - **logback-classic：功能模块，它完整实现了slf4j API**

目录

Contents

- 异常
- 日志技术
  - 日志是什么
  - 日志技术体系、Logback概述
  - Logback快速入门
  - Logback配置详解-输出位置、格式设置
  - Logback配置详解-日志级别设置
- 阶段项目实战

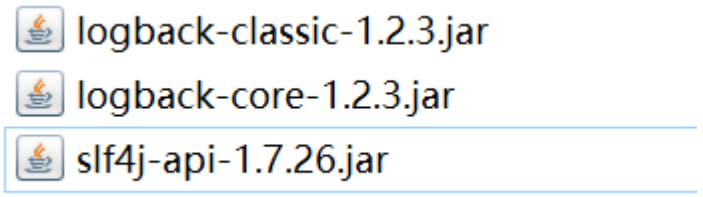
步骤

Logback快速入门

目的：使用Logback日志框架，纪录系统的运行信息。

实现步骤：

①：导入Logback框架到项目中去。在项目下新建文件夹lib，导入Logback的jar包到该文件夹下

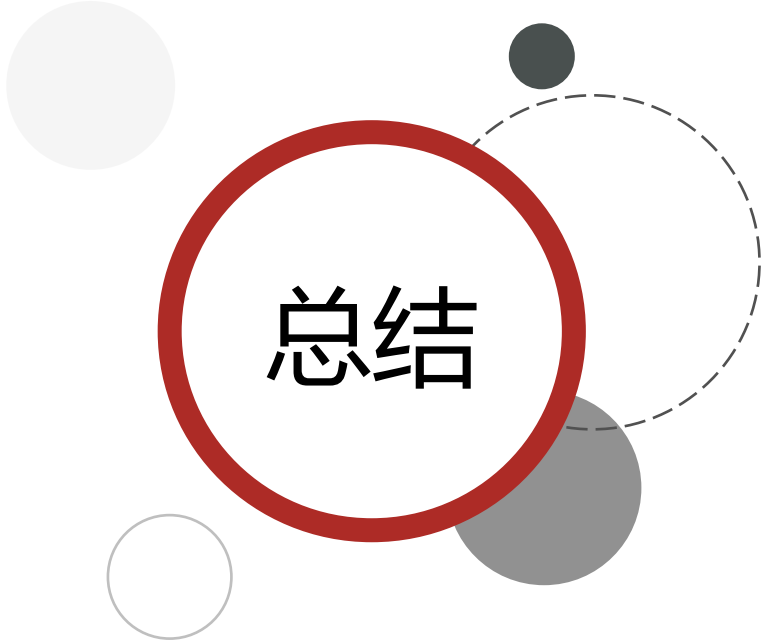


②：将存放jar文件的lib文件夹添加到项目依赖库中去。

③：将Logback的核心配置文件logback.xml直接拷贝到src目录下（必须是src下）。

④：创建Logback框架提供的Logger日志对象，后续使用其方法记录系统的日志信息。

```
public static final Logger LOGGER = LoggerFactory.getLogger( "类名");
```



1. 使用Logback的开发步骤是怎么样的?

①：在项目下新建文件夹lib，导入Logback的相关jar包到该文件夹下，并添加到项目库中去。

②：必须将Logback的核心配置文件logback.xml直接拷贝到src目录下。

③：在代码中获取日志的对象

④：调用日志对象的方法记录日志信息

目录

Contents

对Logback日志框架的控制，都是通过核心配置文件logback.xml来实现的。

Logback日志输出位置、格式设置：

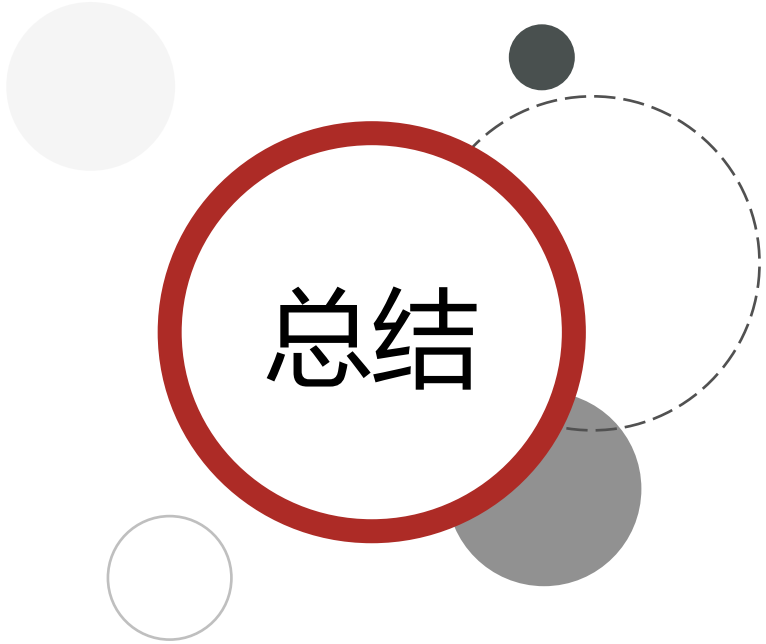
- 通过logback.xml 中的<append>标签可以设置输出位置。
- 通常可以设置2个日志输出位置：一个是控制台、一个是系统文件中

输出到控制台的配置标志

```
<appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
```

输出到系统文件的配置标志

```
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
```



1. 在核心配置文件Logback.xml中可以配置的日志方向有哪些？

输出到控制台的配置标志

```
<appender name="CONSOLE“ …
```

输出到系统文件的配置标志

```
<appender name=“FILE” …
```

目录

Contents

1、如果系统上线后想关闭日志，或者只想记录一些错误的日志信息，怎么办？

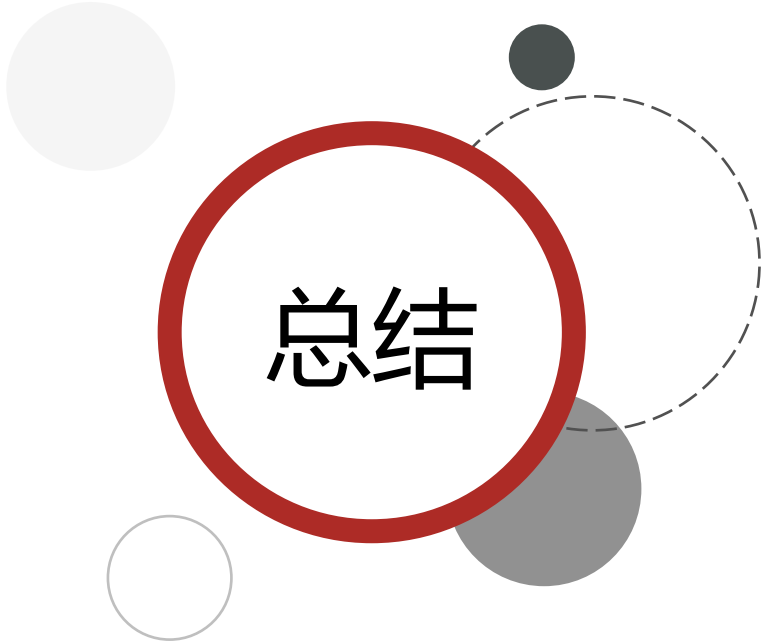
- **可以通过设置日志的输出级别来控制哪些日志信息输出或者不输出。**

日志级别

- ALL 和 OFF分别是打开、及关闭全部日志信息。
- 除此之外，日志级别还有： TRACE < DEBUG < INFO < WARN < ERROR ；默认级别是DEBUG，对应其方法
- 作用：当在logback.xml文件中设置了某种日志级别后，系统将只输出当前级别，以及高于当前级别的日志。

具体在<root level=“INFO”>标签的level属性中设置指定系统的日志级别。

```
<root level=“INFO">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE" />
</root>
```



1、设置日志输出级别的作用是什么？

- **用于控制系统中哪些日志级别是可以输出的。**

2、Logback的日志级别是什么样的？

- **ALL 和 OFF分别是打开全部日志和关闭全部日志**
- **级别程度依次是：TRACE< DEBUG< INFO<WARN<ERROR**
- **默认级别是debug（忽略大小写），只输出当前级别及高于该级别的日志**

目录

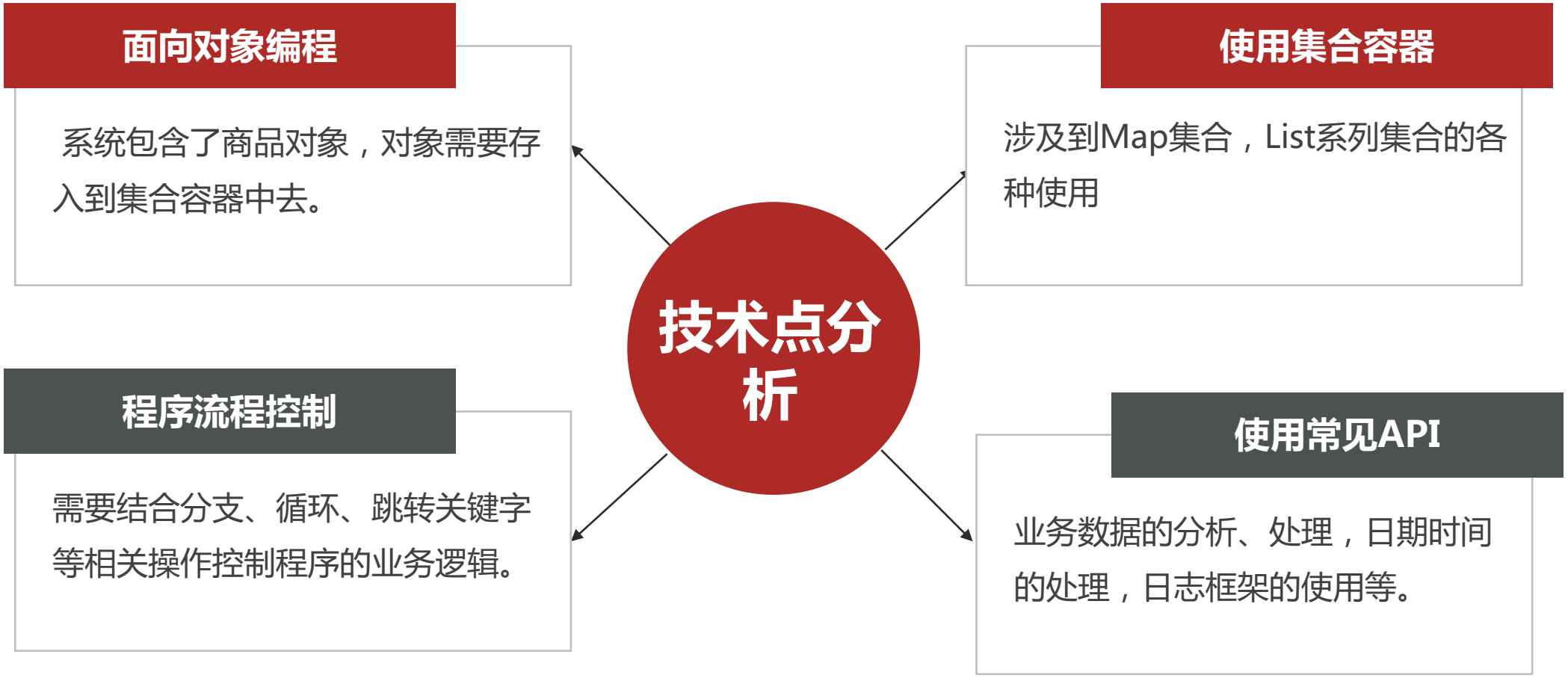
Contents

- 异常
- 日志技术
- 阶段项目实战
  - 商品管理系统简介、项目功能演示
  - 日志框架搭建、系统角色分析
  - 首页设计、退出系统、商品信息展示
  - 商品上架
  - 商品下架
  - 修改商品信息
  - 查询商品信息（练习）

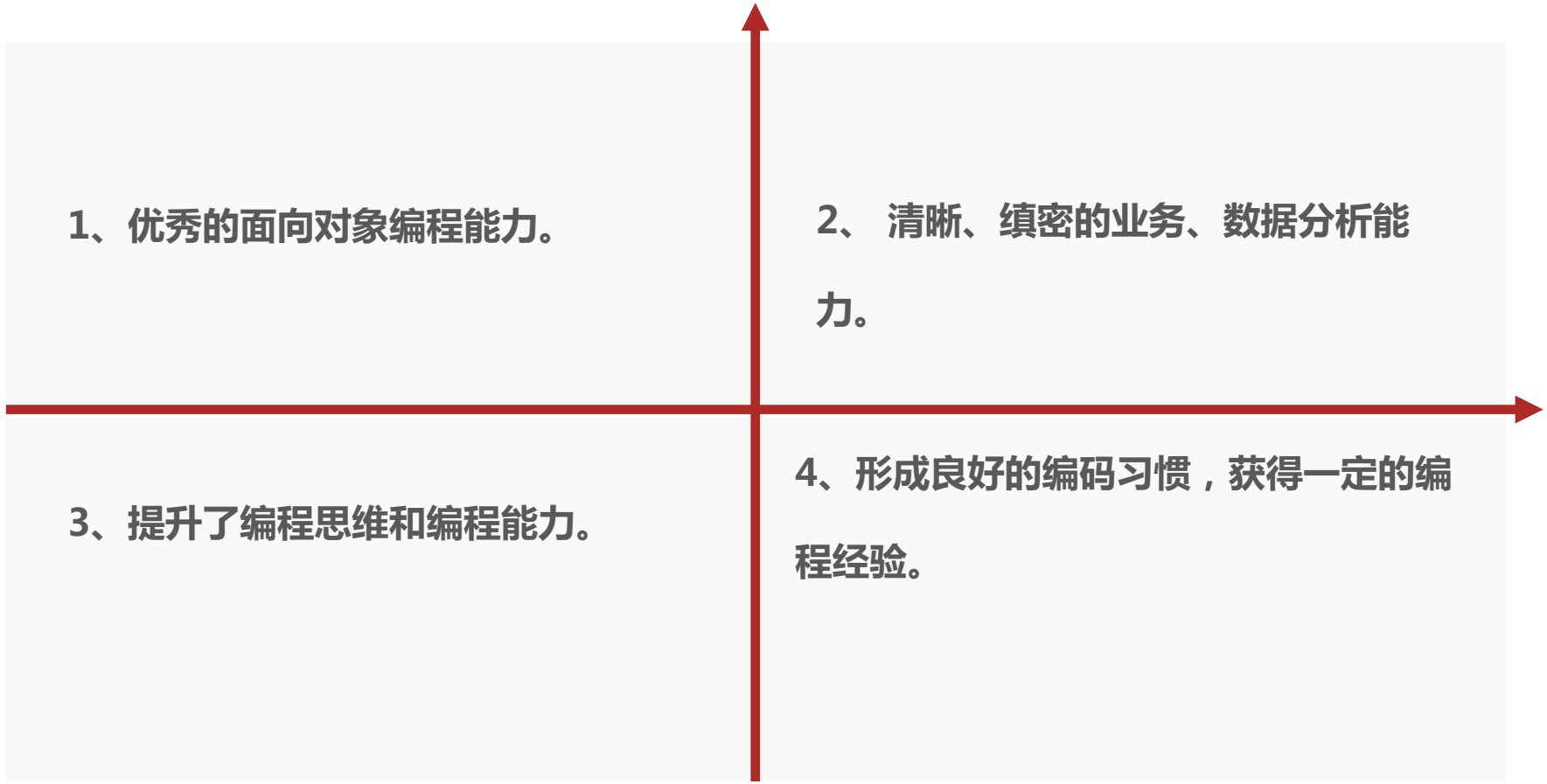
商品管理系统功能演示


```
==================欢迎进入黑马商城系统==============
1、查看全部商品信息
2、上架商品信息
3、下架商品信息
4、修改商品信息
5、查询商品信息
6、退出系统
请您输入命令：
```

商品管理系统技术选型分析：



学习本项目，你将至少得到如下收获：



目录

Contents

步骤

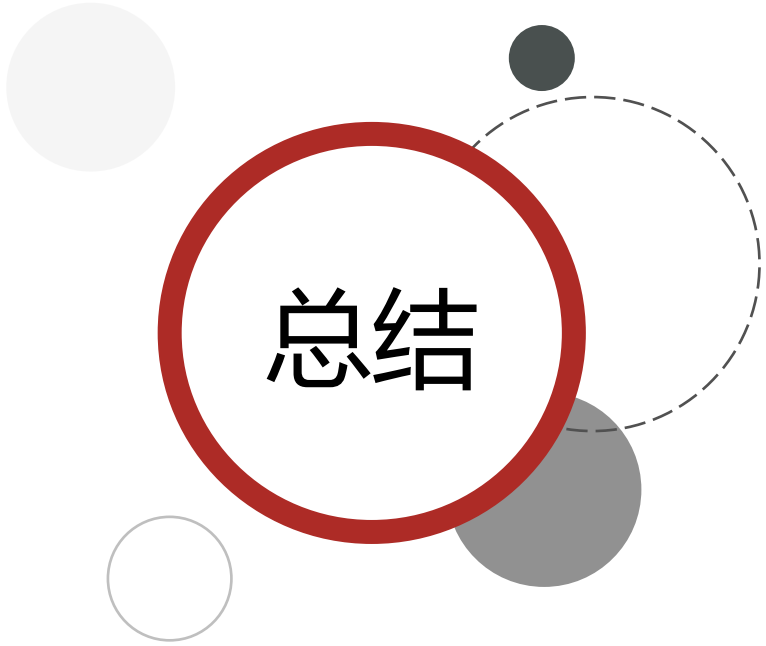
日志框架搭建、系统对象设计

① 集成日志框架、用于后期记录日志信息。

② 定义一个商品类，用于后期创建商品对象，封装商品信息。

③ 定义一个静态的Map集合用于存储商品，键是店铺名称，值是其商品橱柜。

```
店铺：传智教育
        商品名称：《干就对了，Java入门》
        商品价格：30.0
        商品类型：书籍
        商品库存：10000
        商品描述：作者：dlei
        商品上架时间：2021年11月04日 10:25:06 周四 上午
        ---------------------------
        商品名称：《MySQL史上最全手册》
        商品价格：10.0
        商品类型：书籍
        商品库存：99
        商品描述：详细介绍如何避免删库跑路
        商品上架时间：2021年11月04日 10:25:06 周四 上午
```



1、商品信息准备如何进行封装？

- **定义了Article商品类，后期创建商品对象封装商品信息。**

2、系统准备如何存储全部店家和其商品信息的？

- **定义一个静态的Map集合用于存储的，键是店铺名称，值是其商品橱柜。**
- **Map<String,List<Article>>**

目录

Contents

- 日志框架
- 阶段项目实战
  - 商品管理系统简介、项目功能演示
  - 日志框架搭建、系统角色分析
  - 首页设计、退出系统、商品信息展示
  - 商品上架
  - 商品下架
  - 修改商品信息（练习）
  - 查询商品信息（练习）

步骤 首页展示、退出系统、商品详情展示

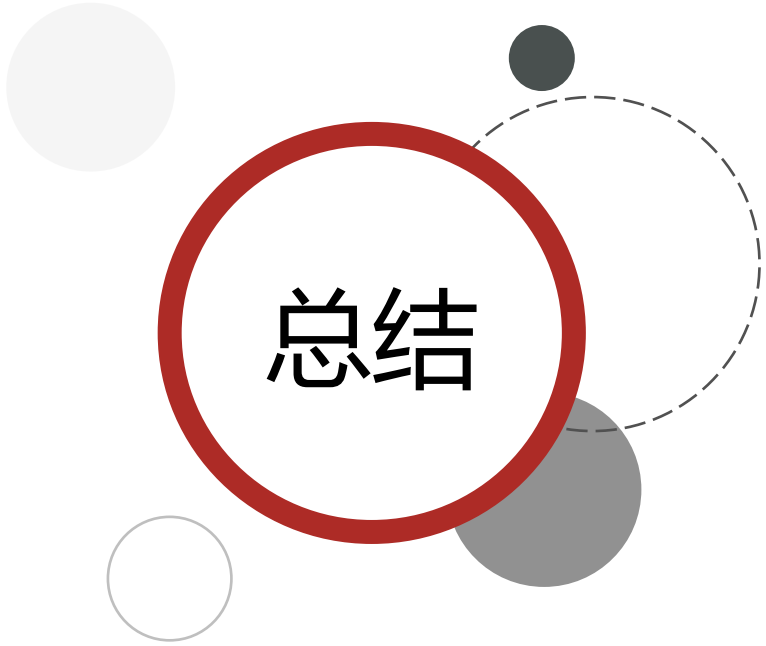
① 首页展示：展示命令，用switch分支判断用户的选择。

② 展示系统全部店家、商品详情：必须展示成指定格式。

③ 退出功能：就是干掉当前程序。

```
==========欢迎进入黑马商城系统==========
1、查看全部商品信息
2、上架商品信息
3、下架商品信息
4、修改商品信息
5、查询商品信息
6、退出系统
请您输入命令：
1
==========商品展示页面==========
店铺：传智教育
        商品名称：《干就对了，Java入门》
        商品价格：30.0
        商品类型：书籍
        商品库存：10000
        商品描述：作者：dlei
        商品上架时间：2021年11月04日 12:02:01 周四 下午
        ---------------------------
        商品名称：《MySQL史上最全手册》
        商品价格：10.0
        商品类型：书籍
        商品库存：99
        商品描述：详细介绍如何避免删库跑路
        商品上架时间：2021年11月04日 12:02:01 周四 下午
        ---------------------------
店铺：黑马生发旗舰店

        商品名称：脱发喷雾
        商品价格：78.5
        商品类型：药品
        商品库存：666
        商品描述：秃顶看过来
        商品上架时间：2021年11月04日 12:02:01 周四 下午
        ---------------------------
```



1、系统是怎样展示出全部店家和商品信息的？

- **遍历存储店家和商品的Map<String,List<Article>> 集合。**

目录

Contents

步骤 商品上架

① 商品上架：就是指定店铺，添加一个新的商品信息给它。

② 如果店铺存在：则给其添加一个新的商品对象。

③ 如果店铺不存在：则添加一个新店铺，并给其添加一个新的商品对象。

总结

1、商品上架是如何实现的，简述一下实现的思路？

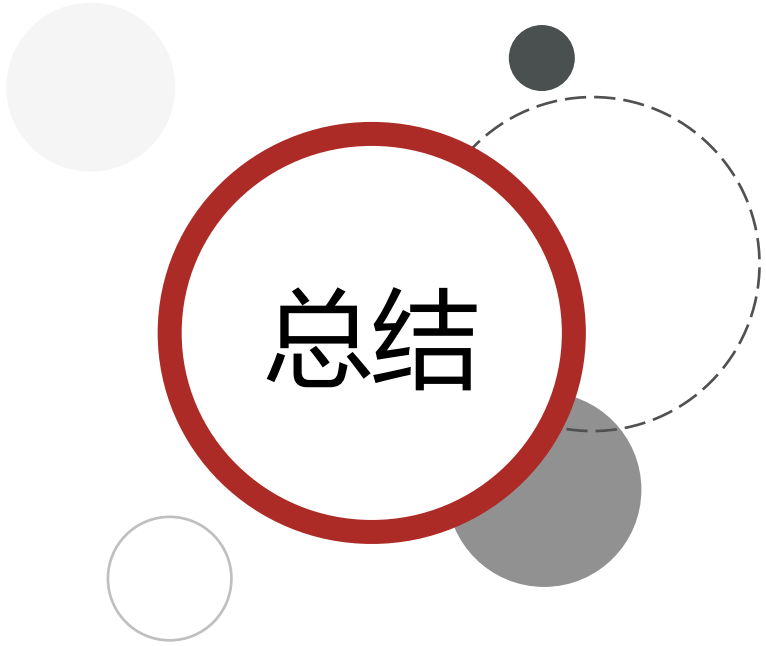
- **让用户输入店铺名称，判断店铺是否存在。**
- **店铺存在，或者其橱柜**
- **店铺不存在，创建新橱柜，加入店铺和橱柜到Map集合中去**
- **为店铺的橱柜添加一个新的商品对象。**

目录

Contents

步骤 商品下架

① 商品下架：就是指定店铺，并指定一个商品，从该店铺的橱柜中删除该商品。



1、商品下架是如何实现的，简述一下实现的思路？

- **先让用户输入店铺，获取其橱柜。**
- **如果橱柜存在，再让用户录入商品名称。**
- **根据商品名称和店铺名称查询出该商品对象。**
- **再从橱柜中删除该商品对象即可。**

目录

Contents

传智教育旗下高端IT教育品牌