

工行学习笔记

Java

Java的文件结构 (20221101)

一个完整的Java源程序应该包含以下内容：

1. package 语句
2. import 语句
3. public class 类

Java的命名规则 (20221101)

Java的包名

全小写，中间用点分隔开。例如：java.awt.event。

Java的类名、接口名

首字母大写。例如：Class HelloWorldApp，Interface Collection。

Java的方法名与变量名

首字母小写，后面的单词首字母大写。例如：isButtonPressed。

Java的常量名

全大写。例如：int YEAR。

Java的常用语法和术语

静态类 (**)

内部类 (嵌套类) (20221111)

即定义在类内的类。

非静态内部类与静态内部类

非静态内部类需要创建外部类对象，对其访问。

静态内部类不需要创建外部类对象，可以直接访问。

匿名对象与匿名类 (20221114)

匿名对象

匿名对象是指在创建对象时，只有创建对象的语句，却没有把对象地址赋值给某个变量。

例如：

创建一个普通对象：Person P = new Person();

创建一个匿名对象：new Person();

匿名对象具有如下特点：

1. 创建匿名对象可以直接使用，没有变量名。例如：new Person().work();
2. 匿名对象在没有指定其引用变量时，只能使用一次。
3. 匿名对象可以作为方法接收的参数，方法返回值使用。

匿名类

匿名类是指没有类名的内部类，必须在创建时使用 new 语句来声明类。

```
new <类或接口> () {  
    // 类的主体  
};
```

这种形式的 new 语句声明一个新的匿名类，它对一个给定的类进行扩展，或者实现一个给定的接口。

1. 继承一个类，重写其方法。
2. 实现一个接口（可以是多个），实现其方法。

钻石操作符（20221111）

即<>。

java7之前：List<String> list = new ArrayList<String>();

java7的变化：可以进行类型推断, List<String> list = new ArrayList<>();**

Java的抽象类和抽象方法（20221017）

抽象类的语法：

```
( abstract ) class ( class_name ) {  
    ( abstract ) ( type ) ( method_name ) ( parameter-list ) ;  
}
```

抽象方法的特征：

抽象方法只有声明，但是没有实现。

abstract关键词只能用于普通方法，不能用于static方法和构造方法。

抽象方法没有方法体。

抽象方法必须在抽象类中。

子类重写时，必须重写全部的抽象方法。

不能用private修饰，因为需要重写。

抽象类的特征：

抽象类和抽象方法都要使用 abstract 关键字声明。

抽象类中可以有抽象方法或者具体方法。

抽象类不能实例化。

Java接口（20221017）

接口的声明：

```
[可见度] interface 接口名称 [extends 其他的接口名] {  
    // 声明变量  
    // 抽象方法  
}
```

接口的特征：

接口中每一个方法也是隐式抽象的，接口中的方法会被隐式的指定为 public abstract (只能是 public abstract)。

接口中可以含有变量，但是接口中的变量会被隐式的指定为 public static final 变量 (并且只能是 public static final)。

接口中的方法是不能在接口中实现的，只能由实现接口的类来实现接口中的方法。

接口的实现

接口实现的语法：

```
[class] [class_name] implements 接口名称[, 其他接口名称, 其他接口名称..., ...] ...
```

当类实现接口的时候，类要实现接口中所有的方法。否则，类必须声明为抽象的类。

一个类可以同时实现多个接口。

类在实现接口的方法时，不能抛出强制性异常，只能在接口中，或者继承接口的抽象类中抛出该强制性异常。(没太看懂)

接口的继承

接口的多继承

抽象类与接口的相似性与差异 (20221114)

抽象类与接口的相似性

1. 接口和抽象类都不能被实例化，它们都位于继承树的顶端，用于被其他类实现和继承。
2. 接口和抽象类都可以包含抽象方法，实现接口或继承抽象类的普通子类都必须实现这些抽象方法。

抽象类与接口的差异

1. 接口里只能包含抽象方法，静态方法和默认方法，抽象类则完全可以包含普通方法。
2. 抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是 public static final 类型。
3. 接口不能包含构造器和初始化块，抽象类可以包含构造器和初始化块。
4. 一个类只能继承一个抽象类，而一个类却可以实现多个接口。

什么时候使用抽象类和接口

1. 如果你拥有一些方法并且想让他们中的一些有默认实现，那么使用抽象类。
2. 如果你想实现多重继承，那么你必须使用接口。Java不支持多继承，但类可以实现多个接口。
3. 如果基本功能在不断改变，那么就需要使用抽象类。如果不断改变基本功能并且使用接口，那么就需要改变所有实现了该接口的类。

构造器和初始化块(20221116)

构造器

即构造函数。

初始化块

它也是类的一种成员，修饰符只能用static修饰，被修饰称为静态初始化块，也可以不修饰。

实际上在编译java块后，初始化块会消失，初始化块的代码会被“还原”到构造器中，且位于构造器代码前面。

示例：

```
public class Test{  
  
    static {  
  
        a = 6; }  
  
}
```

Java关键字

关键字default (20221018)

Java 8 以后的新功能。

如果不使用public、private和protected修饰，那么默认就是 default 访问权限。default 被称为包访问，因为该权限下的资源可以被同一包（库组件）中其他类的成员访问。

在接口中可以通过default关键词定义默认方法，无需类进行实现。

关键字super (20221114)

子类不能继承父类的构造方法，因此，如果要调用父类的构造方法，可以使用 super 关键字。super 可以用来访问父类的构造方法、普通方法和属性。

super 关键字的功能：

1. 在子类的构造方法中显式的调用父类构造方法
2. 访问父类的成员方法和变量。

关键字static (20221114)

static变量

static 关键字用来声明独立于对象的静态变量，无论一个类实例化多少对象，它的静态变量只有一份拷贝。

static方法

static 关键字用来声明独立于对象的静态方法。静态方法不能使用类的非静态变量。

关键字 final (20221114)

被final修饰的变量一旦赋值后，不能被重新赋值。

final 通常和 static一起使用来创建类常量。

关键字 instanceof (20221115)

严格来说 instanceof 是 Java中的一个双目运算符。可以用于判断一个对象是否为一个类的实例。

语法：boolean result = obj instanceof Class 其中，obj是一个对象，Class是一个类的名称。

instanceof的用法如下：

1. 声明一个 class 类的对象，判断 obj 是否为 class 类的实例对象。

例如：

```
Integer integer = new Integer(1);  
System.out.println(integer instanceof Integer);
```

2. 声明一个 class 接口实现类的对象 obj，判断 obj 是否为 class 接口实现类的实例对象。

例如：

Java 集合中的 List 接口有个典型实现类 ArrayList。

```
public class ArrayList extends AbstractList implements List, RandomAccess, Cloneable,  
java.io.Serializable
```

我们就可以用 instanceof 来进行判断：

```
ArrayList arrayList = new ArrayList();  
System.out.println(arrayList instanceof List);
```

3. obj 是 class 类的直接或间接子类，可以用于判断其实际的类名。

Java数据类型（20221018）

引用类型

Java中数据类型分为基本数据类型和引用数据类型。

引用的本质，保存在栈上的地址，指向堆上的对象。

Java泛型（20221018）

泛型，参数化类型。

泛型类

```
public class 类名 <泛型类型1, ...> { }
```

其中泛型类型必须用引用类型。

泛型方法

```
class Demo{  
    public T fun ( T t ) { // 可以接收任意类型的数据  
        return t ;    // 直接把参数返回 }  
};  
  
public class GenericsDemo26{  
    public static void main ( String args[] ) {  
        Demo d = new Demo ( ) ; // 实例化Demo对象  
        String str = d.fun ( "汤姆" ) ; // 传递字符串  
        int i = d.fun ( 30 ) ; // 传递数字，自动装箱  
        System.out.println ( str ) ; // 输出内容  
        System.out.println ( i ) ; // 输出内容 }  
};
```

泛型接口

Java中的参数传递（20221018）

Java中引用类型的本质，是保存在栈上的地址，指向堆上的对象。

Java中的参数传递严格意义上说应该是按共享传递。

按共享传递，是指在调用函数时，传递给函数的是实参的地址的拷贝（如果实参在栈中，则直接拷贝该值）。

Java的Class对象 (20221019)

java有两种对象：实例对象和Class对象。

有三种获得Class对象的方式：

1. Class.forName (“类的全限定名”)：这种方法不需要实例对象。
2. 实例对象.getClass ()
3. 类名.class (类字面常量)：这种方法不会初始化。

Java的Class文件

Java的集合框架

在Java中，要引入 java.util包才能使用集合框架。

List接口 (20221115)

语法：List list1 = new ArrayList<>(); 其中，T是列表的数据类型。

Maven仓库

POM文件 (20221111)

常用标签结构

parent标签

```
xxx
xxx
xxx
xxx
```

必要标签 (最小配置标签，也就是一个POM文件必须配置的四个标签)

```
4.0.0
xxx
xxx
1.0-SNAPSHOT
```

常用标签

```
jar
xxx-maven
http://maven.apache.org
A maven project to study maven.
```

value

依赖标签

junit
junit
3.8.1
test

构建

maven-compiler-plugin

1.6

其他

.....
.....
.....
.....
.....
.....

Super POM

无论 POM 文件中是否显示的声明，所有的 POM 均继承自一个父 POM，这个父 POM 被称为 Super POM，它包含了一些可以被继承的默认设置。

Java注解和反射

常用注解

@Override 重载

@Deprecated 过时

@SuppressWarnings 关闭警告

自定义注解

注解的属性值类型可以为以下三种之一：

- 基本类型
- 数组类型
- 枚举类型

定义注解：

```
@interface Author {  
    int value();  
    String name();  
    String[] phoneNumber();  
}
```

使用注解：

```
@Author(value = 666,name = "goldsunC",phoneNumber = "123456789")  
class Student {}
```

任何注解应该不影响程序代码的执行，无论增加、删除注解，代码都应该正常执行（存疑）

反射

获得类的对象

1. .class 属性 示例：Class classinfo = String.class;
2. getClass方法 示例：String str = "123"; Class classinfo = str.getClass();
3. Class.forName("类名")方法 示例：Class classinfo = Class.forName("java.lang.String");

获得类的信息

利用反射创建对象并调用方法

获得类的注解

Java错误

错误 和 异常

这是根据Java Api的角度进行分类的

JSP

JSP简介

全称：Java Server Pages。中文名称：java服务器页面。本质是一个简化的 Servlet 设计。
JSP是传统的HTML文件中插入Java程序段（Scriptlet）和JSP标记（tag），从而形成JSP文件。

脚本程序 (20221011)

语法 : <% %>

例子 :

JSP 声明 (20221011)

语法 : <%! %>

例子 :

<%! int i = 0 ; %>

<%! int a , b , c ; %>

<%! Circle a = new Circle (2.0) ; %>

JSP 表达式 (20221011)

语法 : <%= %>

JSP 注释 (20221011)

语法 : <%-- 注释--%>

JSP 指令 (20221011)

JSP指令用来设置JSP页面相关属性的设置

page指令

例子 : <%@ page language="java" contentType="text/html ; charset=GB18030" pageEncoding="GB18030" %>

等价语法 : <jsp : directive.page attribute="value" />

page 标签的 attribute :

attribute 标签	用途
buffer	指定out对象使用缓冲区的大小
autoFlush	控制out对象的缓存区
contentType	指定当前JSP页面的 MIME类型和字符编码
errorPage	指定当 JSP 页面发生异常时常要转向的错误处理页面
isErrorPage	指定当前面是否可以作为另一个JSP页面的错误处理页面
extends	指定 servlet 从哪一个类继承
import	导入要使用的 Java类
info	定义 JSP 页面的描述信息
isThreadSafe	指定对JSP页面的访问是否为线程安全
language	定义 JSP 页面所用的脚本语言，默认是 Java
session	指定 JSP 页面是否使用 session
isElIgnored	指定是否执行EL表达式

isThreadSafe 指定对JSP 页面的访问是否为线程安全 language 定义 JSP 页面所用的脚本语言，默认是 Java session 指定 JSP 页面是否使用 session isElIgnored 指定是否执行 EL表达式 scriptingEnabled 确定脚本元素能否被使用

include指令

include 标签指定其他文件

例子：<%@ include_file="/Templates/fh_head.jsp" %>

taglib指令

<%@ taglib %>引入标签库的定义，可以是自定义标签。

例子：<%@ taglib_uri="/WEB-INF/ssts.tld" prefix="ssts"%>

JSP 行为 (20221011)

JSP 行为标签使用XML 语法结构来控制 servlet 引擎，只有XML 这一种语法格式。

语法：<jsp: action_name attribute = "value" />

action_name 标签选项：

action_name 标签选项	用途
jsp : include	用于在当前页面中包含静态或动态资源
jsp : useBean	寻找和初始化一个JavaBean组件
jsp : setProperty	设置 JavaBean组件的值
jsp : getProperty	将 JavaBean组件的值插入到 output 中
jsp : forward	从一个JSP文件向另一个文件传递一个包含用户请求的request对象
jsp : plugin	用于在生成的 HTML 页面中包含Applet 和JavaBean对象
jsp : element	动态创建一个XML元素
jsp : attribute	定义动态创建的XML元素的属性
jsp : body	定义动态创建的XML元素的主体
jsp : text	用于封装模板数据

通配符与正则表达式

通配符（20221018）

通配符是由shell解析的，一般用来匹配文件名

- * 匹配任何字符
- ? 匹配任意一个字符
- [] 匹配中括号里的任意指定的一个字符，但只匹配一个字符

正则表达式（20221018）

正则表达式用来处理文本里的内容

基础正则

- 前一个字符匹配0次或任意多次
 - ． 匹配除了换行符以外任意一个字符
 - ^ 匹配行首。例如：^helloworld会匹配以helloworld开头的行
 - \$ 匹配行尾。例如：helloworld\$ 会匹配以helloworld结尾的行
 - [] 匹配中括号里的任意指定的一个字符，但只匹配一个字符
 - \ 转义符，取消特殊含义
 - {n} 表示其前面的字符恰好出现n次
 - {n,} 表示其前面的字符出现不小于n次
 - {n, m} 表示其前面的字符至少出现n次，最多出现m次

转义字符表

Jenkins

Jenkins 安装 (2022居家办公期间)

war安装

- 1. java -jar Jenkins.war --httpPort=[指定端口]

docker安装

- 1. 拉取docker镜像：docker pull jenkins/jenkins:lts
- 2. 启动 docker run --name jenkins --privileged=true -p 8081:8080 -p 50001:50000 -u root -d -v /home/czpfreedom/jenkins/jenkins_jdk/java-1.17.0:/usr/local/java -v /home/czpfreedom/jenkins/jenkins_maven:/usr/local/maven -v /home/czpfreedom/jenkins/jenkins_home:/var/jenkins_home -v /run/docker.sock:/run/docker.sock jenkins/jenkins:lts
- 3. 进入容器： docker exec -it [container_id] sh
- 4. 查看密码： cat /var/jenkins_home/secrets/initialAdminPassword
- 5. 在localhost界面输入密码，完成安装

Jenkins凭证 (20221223)

git需要ssh的公钥认证，才能拉取仓库。
一种可行的操作是：在本机创建ssh密钥对后，将公钥存放在gitlab，而在jenkins凭证中保存私钥。

Jenkinsfile (20221223)

Gradle和Groovy

其他

- 1. 最好用chrome操作jenkins，IE可能会出现一些难以理解的bug。

Jenkins插件

内网安装插件方法：

在<https://updates.jenkins.io/current/update-center.json>网站找到该插件的URL，然后下载插件，然后上传该插件

Jenkins域用户登录

- 1. 安装Active Directory插件
- 2. 全局安全配置中设置安全域为Active Directory

Docker

docker

Option	解释	用法
--privileged	使用该参数，container内的root拥有真正的root权限	--privileged=true

docker ps

Option	解释	用法
-a	展示所有容器，包括	

docker run

Option	解释	用法
-d	后台运行容器，并返回容器ID	
-i	以交互模式运行容器，通常与 -t 同时使用	
-t	为容器重新分配一个伪输入终端，通常与 -i 同时使用	
-p	指定端口映射	-p 主机端口:容器端口
-e	设置环境变量	-e username=[用户名]
--env-file	从指定文件读入环境变量	--env-file=[文件名]

docker start

Html

Html的内联元素与块级元素（20221019）

如：

第一第二第三

第四

第五

第六

是内联的，

是块级的

这种分类也不应该和CSS中的同名术语混淆。

HTML5 重新定义了元素的类别：见 元素内容分类（译文）。

Html的属性（20221019）

建议属性的值加上引号。

Eclipse

Eclipse的常用快捷键（20220929）

总览

快捷键	IDEA	Eclipse
全局搜索Class/File	Ctrl/Command+N	Ctrl+Shift+R/T
查看继承关系	Ctrl/Command+H	Ctrl+T
constructor/getter-setter/toString	Alt+Insert	Alt+Shift+S
回退	Ctrl/Command+Alt+方向左键	Alt+方向左键
补全声明	Ctrl/Command+Alt+V	Alt+/
批量导包	Ctrl/Command+Enter	Ctrl+Shift+O
格式化代码	Ctrl/Command+Alt+L	Ctrl+Shift+F
查看指定结构使用过的地方	Alt+F7	Ctrl+Alt+
快速修复	Alt+Enter	Ctrl+1
全局查找文本内容	Ctrl/Command+Shift+F	Ctrl+H

ctrl+shift+r :

open resource , 打开资源。

它可以打开当前eclipse的工作区中所有（打开的）工程中所有类型的文件，但只限手动编写的文件不含工程中引用到的jar包中的类、接口。

ctrl+shift+t :

open type , 打开类型。

它可以打开当前eclipse的工作区中所有（打开的）工程中所有java文件，包括jar包中的类和接口。

Spring Boot

Spring Boot 项目手动搭建(20221110)

1. 搭建Maven仓库 配置Maven的Settings.xml
2. 创建Springboot项目
3. pom.xml parent指定版本2.0.3
4. 如果出现No goals have been specified for this build错误，需要在标签添加compile
5. 如果出现 springboot 程序包org.junit.jupiter.api不存在，是因为java1.8版本问题，可以删除对应的test文件

Springboot结构

Springboot 注解

控制层

@Controller

@RequestMapping

```
@RequestMapping(  
    path = { "/test" },  
    params = { "name", "userId"},  
    method = { RequestMethod.GET},  
    consumes = {"text/plain", "application/"},  
    produces = "text/plain",  
    headers = "content-type=text/"  
)
```

@ResponseBody

正常情况下，Controller类中函数返回是一个模板对象（页面），但是有时候我们希望返回值数值型（字符串或者Json格式数据），这时就需要添加@ResponseBody注解

@RestController

@RestController=@ResponseBody+@Controller

@GetMapping

@GetMapping=@RequestMapping (method=RequestMethod.GET)

@PostMapping

@PostMapping=@RequestMapping (method=RequestMethod.POST)

@Value

Python

指定镜像源(20221109)

示例：python get-pip.py -i <http://pypi.douban.com/simple/> --trusted-host pypi.douban.com

可用镜像源：<http://mirrors.aliyun.com/pypi/simple/>

pip命令

Python-Mysql

Win7安装MySQLdb(20221104)

1. 下载MySQLdb 1.2.3 for Windows and Python 2.7 64 bit

下载地址：<https://www.codegood.com/downloads>

2. 如果显示python2.7 required 错误，需要在注册表中添加。
3. 创建 import MySQLdb 脚本，确定该包已经被成功安装。

`_mysql`

用法：`from MySQLdb import _mysql`

`connect`

获得一个对Mysql/服务端的连接。

用法：

```
db=_mysql.connect(host="localhost",user="joebob",passwd="moonpie",db="thangs")
```

```
db=_mysql.connect(host="outhouse",db="thangs",read_default_file=~/.my.cnf")
```

`query`

用法：`db.query("""SELECT spam, eggs, sausage FROM breakfast WHERE price < 5""")`

`store_result()`

`use_result()`

Python-Sqlalchemy (20221117)

基本用法 (20221117)

加载模块

```
from sqlalchemy.ext.declarative import declarative_base
```

```
from sqlalchemy.orm import sessionmaker
```

```
from sqlalchemy import Column, String, create_engine
```

声明性表格，创建自定义类

我们可以通过继承Base类创建自定义类

```
BASE = declarative_base()
```

```
class User(BASE):
```

```
    __tablename__ = 'users'
```

```
    id = Column(Integer, primary_key=True)
```

```
    name = Column(String)
```

```
    fullname = Column(String)
```

```
    password = Column(String)
```

其相当于：

```
Table(
    "user",
    Base.metadata,
    Column("id", Integer, primary_key=True),
    Column("name", String),
    Column("fullname", String),
    Column("nickname", String),
)
```

我们可以通过`user_table = User.__table__`获取这个自定义的类本身。

连接数据库

```
MySQLEngine=create_engine('mysql+mysqldb://root:123@localhost:3306/test?charset=utf8',  
encoding='utf-8', echo=True)
```

创建表和删除表

```
Base.metadata.create_all(MySQLEngine)
```

```
Base.metadata.drop_all(MySQLEngine)
```

创建一个会话

```
MySQLSession = sessionmaker(bind=MySQLEngine)
```

```
session = MySQLSession()
```

```
//会话操作
```

```
session.close()
```

插入数据

```
Stu = Student(sname='张三', sno='2016081111')
```

```
session.add(Stu)
```

查询数据

```
Stu = session.query(Student).filter(Student.sno == '2016081111')
```

```
print ( Stu[0].sname )
```

修改数据

```
Stu = session.query(Student).filter(Student.sno == '2016081111').first()
```

```
Stu.sname = '李华'
```

```
session.commit()
```

删除数据

```
session.query(Student).filter(Student.sno == '2016081111').delete()
```

```
session.commit()
```

常用数据类型和参数 (20221118)

常用数据类型

使用时需要导入。

```
from sqlalchemy import Column, Integer, String, Float, DECIMAL, Boolean, Enum, Date, DateTime, Time
```

```
from sqlalchemy.dialects.mysql import LONGTEXT
```

Integer : 整形。

Float : 浮点类型 (四舍五入保留 4 为小数)。

DECIMAL : 定点类型 (可规定数据长度及小数位数)。

Boolean : 传递 *True* 、 *False* 进去 , 在数据库中的显示 1 和 0。

enum : 枚举类型 (只能输入 *Enum* 所包含的数值。如 : 男 , 女)。

Date : 传递 *datetime.date()* 进去 , 数据库中存储年月日。

Time : 传递 *datetime.time()* 进去 , 数据库中存储时分秒。

DateTime : 传递 *datetime.datetime()* 进去 , 数据库中存储年月日时分秒 。传入*datetime(2020, 4, 20, 21, 35, 23)*

String : 字符类型 , 使用时需要指定长度 , 区别于 *Text* 类型。

Text : 文本类型。

LONGTEXT : 长文本类型。

Column 常用参数

default : 默认值。

nullable : 是否可空。 *nullable=False* 不允许为空

primary_key : 是否为主键。

unique : 是否唯一 , 存放的内容唯一。

autoincrement : 是否自动增长。

onupdate : 更新的时候执行的函数。例如文件的修改时间 , 就可以在修改了数据后触发执行该函数。

name : 该属性在数据库中的字段名字

query 可用参数

对象属性 :

```
article = session.query(Article).first()
print(article.title)
```

聚合函数

导入包

```
from sqlalchemy import func
```

统计行的数量

```
result = session.query(func.count(Article.id)).first()
print(result)
```

平均值

```
result = session.query(func.avg(Article.price)).first()
print(result)
```

最大值

```
result = session.query(func.max(Article.price)).first()
print(result)
```

最小值

```
result = session.query(func.min(Article.price)).first()
print(result)
```

求和

```
result = session.query(func.sum(Article.price)).first()
print(result)
```

一些自我发掘的可能常用的类和方法 (20221118)

将查询的结果转化为列表：

```
Stu = session.query(Student).filter(Student.sno == '2016081111')
```

```
List_Stu=Stu.all()
```

Python-Tkinter (20221115)

Mysql

常用语法 (20221102)

展示数据库 *show databases;*

选择数据库 *use [database_name];*

创建数据库 *create database [database_name] ;*

删除数据库 *drop database [database_name] ;*

创建数据表 *create table [table_name] ([column_name1] [column_type1] {列级完整性约束1}, [column_name2] [column_type2], , {表级完整性约束1}, {表级完整性约束2});*

约束条件	说明	备注
primary key	标识该属性为该表的主键	可以在列级完整性约束中添加，也可以在表级完整性约束中添加
foreign key	标识该属性为该表的外键	只能在表级完整性约束中添加
not null	取值不能为空	在列级完整性约束中添加
unique	取值唯一	在列级完整性约束中添加
auto_increment	自动增加	在列级完整性约束中添加
default	默认值	在列级完整性约束中添加

删除数据表 *drop table [table_name]*

修改表名 *alter table [table_name] rename to [new_table_name];*

插入数据 *insert into [table_name] ([field1], [field2]) values ([value1], [value2],);*

修改数据 *update [table_name] set [field1]=[new_value1], [field2]=[new_value2] {where []}*

查询数据 *select [column1],[column2] from [table_name];*

删除数据 *delete from [table_name] where [];*

插入字段 *alter table [table_name] add [column_name] [column_type] {after [old_column_name]};*

修改字段类型 `alter table [table_name] modify [column_name] [new_column_type];`

修改字段类型或名称 `alter table [table_name] change [column_name] [new_column_name] [new_column_type];`

删除字段 `alter table [table_name] drop [column_name];`

修改字段默认值 `alter table [table_name] alter [column_name] set default [default_value];`

删除字段默认值 `alter table [table_name] alter [column_name] drop default;`

指定字段取值1 数据类型：`enum("value1","value2",.....)`最多有 2^{64} 个取值。只能取一个值。

指定字段取值2 数据类型：`set("value1","value2",.....)`最多64个取值。可以取多个值。

查看Mysql的数据存储位置 `show global variables like "%datadir%";`

Chrome开发者工具

F12可以进入开发者界面

Elements

检查python版本

自助终端

Pos 机 (20220929)

Pos 机简介

全称Pointofsales，是一种配有条码或OCR码技术的终端阅读器。

POS系统基本原理是先将商品资料创建于计算机文件内，透过计算机收银机联机架构，商品上之条码能透过收银设备上光学读取设备直接读入后（或由键盘直接输入代号）马上可以显示商品信息（单价，部门，折扣...）加速收银速度与正确性。

POS机是通过读卡器读取银行卡上的持卡人磁条信息，由POS操作人员输入交易金额，持卡人输入个人识别信息（即密码），POS把这些信息通过银联中心，上送发卡银行系统，完成联机交易，给出成功与否的信息，并打印相应的票据。

Pos 机组成

通讯接口电路通常由RS232接口，PINPAD接口，IRDA接口和RS485等接口电路组成。RS232接口通常为POS程序下载口，PINPAD接口通常为主机和密码键盘的接口，IRDA接口通常为手机和座机的红外通讯接口。接口信号通常都是由一个发送信号、一个接收信号和电源信号组成。

MODEM板由中央处理模块、存储器模块、MODEM模块、电话线接口组成。首先，POS会先检测/RING和/PHONE信号，以确定电话线上的电压是否可以使用，交换机返回可以号音，POS拨号，发送灯闪动，开始拨号，由通讯协议确定交换机和POS之间的信号握手确认等，之后才开始POS的效据交换，信号通过MODEM电路收发信号；完成后挂断，结束该过程。

智能Pos 终端架构

智能Pos安全风险点

刷机风险

刷机是Android系统经常遇到的问题，刷机后设备的整个软件体系，可以被非法篡改，也就意味着设备上运行的所有软件，都有可能被替换成非法的软件，甚至每个交易流程和界面都可能被伪造。

Root风险

如果终端操作系统被root，意味着攻击者可以获得终端所有硬件设备的控制权，从而可以向终端中随意添加、修改或替换安全固件，直接或间接导致持卡人信息泄露。

恶意软件攻击

如果不能建立统一、安全、可控的应用市场，终端及收单系统可能受病毒、木马等各类恶意软件的攻击。

虚拟键盘PIN输入

虚拟键盘PIN输入是新兴的技术，应严格执行相关安全要求和安全标准，否则有可能被攻击者用于非法获取持卡人PIN等敏感信息（例如通过录屏窃取PIN或非法应用窃取PIN等）。

信息安全风险

仅允许合法终端接入系统、且交易报文信息进行必要的安全保护

智能pos厂商

如联迪A8、itwell A90等。

上线流程

智能POS运营管理平台投产后，APP研发与上线指引流程。

Mis 系统（20200929）

全称：Management Information System，中文名称：管理信息系统。

MIS收单系统主要由商户收银系统、银行金融手柄系统和银行前置系统三部分组成，银行前置系统包括外联前置和综合前置。

移动 Mis 对接

传统 Mis 对接

Pos流程（20220929）

Pos业务流程

以平安保险为例，流程如下：

1. Pos上输入保单号。
2. Pos（拨号、专线、互联网）（核心区、DMZ区GTCC、互联网）→ 分行GTCC → 第三方（查询保单金额）→ GTCC → Pos
3. 客户刷卡 → 总行综合前置 → 主机
4. Pos打印签购单 → GTCC → 第三方 如扣费成功，出保单；如扣费失败，冲正
5. Pos显示交易成功

Pos业务规则

Pos业务规则有两种，具体如下：

1. 扣费成功，不打印签购单 → 通知第三方 → 第三方返回成功 → 打印签购单
扣费成功，不打印签购单 → 通知第三方 → 第三方返回失败（或网络故障超时） → Pos显示失败，不打印签购单
2. 扣费成功，打印签购单，如果失败（或网络故障超时），进行补登交易，反复补登失败，人工干预。

三种Pos接口

Pos机接口

例如：人工柜面外挂Pos机，和收银机通过串口相连。

库接口

例如：按摩医院，和银行合作收单。自助终端不能外挂Pos机，银行卡插进去，即库接口。

api接口

例如：扫码的地方，即api接口。

综合前置（20200929）

综合前置简介

运行于前置机上的综合前营业务平台是一个集中的、多渠道接入的平台。

综合前置组成

综合前置系统分为服务接入、中央核心系统、异步处理控制和服务接出四部分。

异步处理

异步处理模块用来处理对实时性要求不高，不要求信息交互的异步服务，如冲正业务，短消息定制服务。

综合前置的关键库表

综合前置对数据库的操作涉及三类表单：平台配置类、平台参数类和应用交易配置类表单。

GTCG

GTCG简介（20220929）

GTCG通讯前置，隶属中间业务，主要负责COSP/PASS和第三方通讯。

COSP

COSP简介（20220929）

给予总行CTP的IDE，是中间业务的核心，交易的逻辑，数据库的操作。

移动话费流程（20220929）

柜面/网银/自助，客户输入电话号 → GTCG → 第三方 → GTCG → COSP（插表电话号、卡号、金额）→ GTCG → 自助 → GTCG → COSP（上主机）（插流水表）→ GTCG → 自助（交易完成）

Git

个人令牌（信工所时期）

问题描述

Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.

解决方案

1. 进入github: Setting:Developer Setting
2. 申请令牌并记在记事本中
3. 连接本地仓库和远程仓库 `git remote set-url origin https://<your_token>@github.com/.git`
4. `git push origin master`

Windows 系统

常见bug解决方法

您的连接不是私密连接(20221107)

在网页页面直接输入 `thisisunsafe`

CMD(20221110)

设置CMD的编码

在Shell中输入 `chcp 65001`，即可切换UTF_8编码

端口被占用（居家隔离期 20221122）

查找8080端口 `netstat -ano | findstr 8080`

杀掉进程 `taskkill -pid [pid] -f`

Linux系统

端口被占用（居家隔离期 20221122）

查找8080端口 `lsof -i :8080`

计算机网络

Ajax

即 *Asynchronous Javascript And XML*。使用Ajax技术网页应用能够快速地将增量更新呈现在用户界面上，而不需要重载（刷新）整个页面，这使得程序能够更快地回应用户的操作。

业务术语

冲正业务 (20220929)

一笔交易的反交易。

技术上，银联交易用的比较多，比如一笔取款，超起时收不到回应时或其他原因交易不正确，需要对原始取款发一个冲正交易，防止交易失败了，但又下了客户帐。

业务上，原来的会计上叫隔日红字冲正，即对原来所记的帐务，同方向红字记账。

中央核心系统 (20220929)

中央核心系统包括：机构调用接入、中央控制子系统、异步处理子系统、机构调用接出子系统和后台交易监控管理。

分行特色研发 (20220929)

总行交易：直接通过综合前置上送主机，完成相关交易。

分行特色交易：由前置转发中间业务平台，经过中间业务平台与三方系统对接，并通过DSR接口完成交易流程。

VTM (20220929)

全称Video Teller Machine：中文名称：远程视频柜员机。

去朝阳的自助中心交流的时候，说到：

1. 平谷支行把VTM雪藏了，因为怕丢
2. VTM得后端有人，所以总行现在不推行了
3. 除北京以外，其他城市都只有一个智慧网点
4. 便携式VTM，只要不叫VTM，叫什么都行，叫VTM 总行不喜欢

收单 (20220929)

收单是指签约银行向商户提供的本外币资金结算服务。最终持卡人在银行签约商户那里刷卡消费，银行结算。

回单 (20220929)

回单是指乙付钱给甲，甲是收款方，银行出具给甲的单位表示收到此款项并存入相应账户的凭据。

保单 (20221027)

即保险单。保险人与投保人签订保险合同的书面证明。

保险单的主要内容包括：

1. 双方对有关保险标的事项的说明，包括被保险人名称，保险标的的名称及其存放地点或所处状态、保险金额、保险期限、保险费等。
2. 双方的权利和义务，如承担责任和不予承担的责任等。
3. 附注条件，指保险条款或双方约定的其他条件以及保单变更、转让和注销等事项。

Pos签购单 (20221027)

POS签购单是持卡人刷卡消费的凭证，签购单上除注明银联特约商户名称及其编号、POS机机号、消费时间和金额等外，还特别标明所刷银行卡的银行名称和16位卡号数字。

据银联人士介绍，一式两份的签购单，除消费者保留一份作为消费凭证外，另一份由商户回收保管，用于对账、清算和明细查询等。

名词术语

DSR网关 (20221027)

即*Dynamic Source Routing*，基于源路由方式的按需路由协议。

语法糖 (20221018)

语法糖 (*Syntactic sugar*)，也译为糖衣语法，是由英国计算机科学家彼得·约翰·兰达 (*Peter J. Landin*) 发明的一个术语，指计算机语言中添加的某种语法，这种语法对语言的功能并没有影响，但是更方便程序员使用。通常来说使用语法糖能够增加程序的可读性，从而减少程序代码出错的机会。

OCR (20220929)

全称*Optical Character Recognition*。中文名称：光学字符识别。发展初期，60、70年代，识别以数字为主，如邮政编码识别系统。

OCR主要技术如下：

1. 图像输入、预处理
2. 二值化
3. 去噪
4. 校正倾斜
5. 版面分析
6. 字符切割
7. 字符识别
8. 版面恢复
9. 根据上下文的后处理、校对

Ear包 (20221010)

Ear包=war包+配置文件。

war包即Web归档文件，将Web项目打成war包可以直接拷贝到Web服务器发布目录。

信创 (20220929)

即国产化。信创的操作系统是linux (统信UOS)，芯片是飞腾 (ft2000)。

通信UOS (**)

ft2000 (**)

WAP协议 (20221028)

即*Wireless Application Protocol*，无线应用协议。

Tomcat

Tomcat简介 (20220930)

Tomcat是一个免费的开源的 Web应用服务器，属于轻量级服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的首选。

Tomcat本质上就是一个Web服务器和Servlet容器的结合体。

Web 服务器

Web服务器的作用，就是将某个主机上的资源映射为一个URL供外界访问。

Servlet 容器

程序处理请求分为三部分：接受请求、处理请求、响应请求。

接受和响应比较机械，所以打包成了Web服务器。

WAS (20230220)

全称：WebSphere Application Server。

JWT (20221226)

JWT (JSON Web Token) 是目前最流行的跨域认证解决方案，是一种基于 Token 的认证授权机制。

JWT 本质上就是一组字符串，通过 (.) 切分成三个为 Base64 编码的部分：

- **Header**：描述 JWT 的元数据，定义了生成签名的算法以及 Token 的类型。
Header 通常由两部分组成：
 - typ (Type)：令牌类型，也就是 JWT。
 - alg (Algorithm)：签名算法，比如 HS256。
- **Payload**：用来存放实际需要传递的数据
- **Signature (签名)**：服务器通过 Payload、Header 和一个密钥(Secret)使用 Header 里面指定的签名算法 (默认是 HMAC SHA256) 生成。

Signature 部分是对前两部分的签名，作用是防止 JWT (主要是 payload) 被篡改。

这个签名的生成需要用到：

- Header + Payload。
- 存放在服务端的密钥(一定不要泄露出去)。
- 签名算法。

JWT示例

JWT：

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJzdWIiOiIxMjMONTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.  
sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Header：

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Payload：

```
{
  "uid": "ff1212f5-d8d1-4496-bf41-d2dda73de19a",
  "sub": "1234567890",
  "name": "John Doe",
  "exp": 15323232,
  "iat": 1516239022,
  "scope": ["admin", "user"]
}
```

Signature :

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

尚未学习

*java*的注解与反射

*java*的增强*for*循环

*html*的布尔属性

html 元数据

SaaS

ERP

即商户的收银系统。

7.*JSP* 隐含对象

*JSP*支持九个自动定义的变量，称作隐含定义

*MIME*类型

*EL*表达式

session

*MVC*模式

*java*组件

多态