Java

JavaSE基础加强

staic

- static是静态的意思，在内存中只有一份，可以被共享
- 堆内存、栈内存、方法区、代码块（Java类中使用{}为代码块，静态方法块与类一起加载一次，自动触发执行；非静态代码块，每次构建对象的时候，会执行一次）
- 使用场景：表示对象自己的行为的，且方法中需要访问实例成员的，则该方法必须申明成实例方法
- 静态方法只能访问静态成员，实例方法可以访问静态方法

继承（extends）

- 提供代码复用性，增强类的扩展性。共用属性放在父类，独有放在子类。内存分配原理很重要
- 子类不能继承父类的构造器，子类可以直接使用子类名访问父类的静态成员。一个类只能继承一个直接父类
- 子类重写父类的方法会覆盖。重写的方法名称和形参必须一致。
- 子类的构造器会先访问父类的无参数构造器，在执行自己的构造器
- super和this必须放在第一行

权限修饰符、抽象类、接口

- （private、public、protected）final修饰的类不能被继承，final修饰方法不能被重写，final修饰变量只能被赋值一次。static和final一起修饰成员变量相当于变量。
- 常量使用public static final 修饰的成员变量。方便程序维护，提高可读性。常量会有编译优化，在编译阶段进行“宏替换”。
- 修饰符 enum 枚举名称 {}　　枚举第一行罗列枚举内容，建议全大写。
- abstract定义抽象类和抽象方法。抽象方法所在的类必须为抽象类。
- interface新特性。
  - 默认方法：用default修饰，需要用接口实现类的对象来调用
  - 静态方法，用static修饰，默认为public方法，必须用本身的接口名来调用
  - 私有方法，private修饰，jdk9才有，只能在接口内部使用

多态、内部类、 常用API

- 多态形式实现耦合性，便于扩展和维护。

- 内部类可以理解为“寄生”，外部类可以理解为“宿主”。内部类可以访问外部类的私有成员，
- ！匿名内部类。一个没有名字的内部类，写出来会产生一个匿名内部类的对象
- StringBuilder是一个可变字符串类，效率高

时间、包装类

- Java 8在Java.time新增时间和日期的api
  - LocalDate不包含具体时间的API
  - LocalTime：不含日期的时间
  - LocalDateTime:包含日期和时间
  - Instant：代表时间戳
  - DateTimeFormatter 用于做时间的格式化和解析的
  - Duration 计算两个时间的间隔
  - Period 计算两个日期的间隔
- 8中基本类型的引用类型(引用类型相当于基本类型对象化)
  - byte Byte
  - short Short
  - int Integer
  - long Long
  - char Character
  - float Float
  - double Double
  - boolean Boolean

正则表达式

- 字符类
  - [abc] 只能是abc
  - [^abc] 除了abc以外的任何字符
  - [a-zA-Z] a到z A到Z 包括范围
  - [a-d[m-p]] a到d，或m通过p： ([a-dm-p])联合
  - [a-z&&[def]] d,e或f（交集）
  - [a-z]&&[^bc]]a到z，除了b和c：[ad-z]减法

- [a-z&&[^bc]] a到z，除了b和c
- [a-z&&[^m-p]] a到z，除了m到p

- 预定义字符类
  - . 任何字符
  - \d 一个数字：[0-9]
  - \D 非数字: [^0-9]
  - \s 一个空白字符.[\t\n\x0B\f\r]
  - \S 非空白字符: [^\s]
  - \w 英文、数字、下划线t. [a-zA-Z_0-9]
  - \W 一个非单词字符r: [^\w]
- 贪婪的量词
  - X? X, once or not at all
  - X* X, zero or more times
  - X+ X, one or more times
  - Xn{} X, exactly n times
  - Xn{,} X, at least n times
  - Xnm{,} X, at least n but not more than m times

- Lambda：Lambda表达式只能简化函数式接口（只能有一个方法）的匿名内部类的写法形式
- Collection集合、数据结构、List、泛型
  - 集合大小不固定，类型也不固定，集合只能存储引用类型，如果想存储基本类型就采用包装类
  - Collection单列：1.List集合：添加的元素是有序、可重复、有索引的 2.Set集合：添加的元素是无序、不重复、无索引
  - ArrayList底层是基于数组实现的，根据索引块定位元素快，第一次创建并添加第一个元素的时候，在底层创建一个默认长度为10的数组，每次满了会扩张的原来的1.5倍
  - LinkedList底层是双链表，可以完成队列结构和栈结构
  - 并发修改问题：迭代器遍历集合且直接用集合删除元素的时候可能出现。
    增强for循环遍历集合且直接用集合删除元素的时候可能出现。 迭代器遍历集合但是用迭代器自己的删除方法操作可以解决。使用for循环遍历并删除不会存在这个问题
  - 泛型只支持引用类型，

- 定义泛型类：修饰符 class 类名<泛型变量>{} 定义泛型方法：修饰符<泛型变量>反方返回值 方法名称（形参列表）{}

- Map、集合嵌套
  - 双列：键值对集合。map是所有双列集合的祖种
- 不可变集合、Stream流、异常
  - 在list、set、map接口中，都存在of方法，可以创建一个不可变集合。static <E> List<E> of(E...elements)
  - 这种风格将要处理的元素集合看作一种流， 流在管道中传输， 并且可以在管道的节点上进行处理， 比如筛选， 排序，聚合等。
  - 元素流在管道中经过中间操作的处理，最后由最终操作得到前面处理的结果。
  - 异常
    - 常用异常：数组索引越界异常、空指针异常、类型转换异常
    - 自定义异常
- Logback日志框架
  - 日志记录程序运行过程中的信息，并可以进行永久存储
  - 常见的规范：1.Commons Logging 2.Simple Logging Facade For Java
  - 常见框架： Log4J 、Logback（重点）
- File、IO流
  - 相对路径：相对到工程下开始寻找文件
  - 输入流、输出流。字节流、字符流
  - 缓冲流、序列化、打印流、IO框架
    - 缓冲流自带缓冲区，可以提高读写数据的性能
    - 把内存的文件存储到磁盘文件中，称为对象序列化
    - 打印流可以实现方便、高效的打印数据到文件中
    - Properties代表的是一个属性文件，可以把自己对象中的键值对信息存入到一个属性文件中去。可以加载属性文件中的数据到Properties对象中来。
    - Commons IO框架
- 多线程
  - 实现方式：继承Thread类（start和run区别）、实现Runnable接口、实现Callable接口
  - 线程常用方法，getName、setName、currentThread（静态方法）、sleep（静态方法）、run（线程任务方法）、start（线程启动方法）

- 同步代码块：synchronized（同步锁对象）{操作共享资源的代码}　　每次只能一个线程占锁进入访问
- Lock锁
- 线程通信：notify、notifyAll、wait。
- 线程池：可以复用线程的技术（实际应用中每一个对象创建一个线程代价很大）。 实现方法：ExecutorService的实现类ThreadPoolExecutor和Executors调用方法
- 定时器： 一种控制任务延时调用，或者周期调用的技术。Timer、ScheduledExecutorService

网络编程

- 常见的通信模式：CS（Client-Server）、BS（Browser/Server）、
- 网络通信的三要素
  - IP地址：设备再网络中的地址，是唯一的标识
    - IPv4、IPv6（4和6代表字节）
    - 198.168.开头是常用的局域网地址
    - 本机IP：127.0.0.1或者localhost：
    - 常用命令：ipconfi、ping IP地址
    - IP地址操作类：InetAddress
  - 端口：应用程序在设备中唯一的标识
    - 周知端口：0~1023，被预先定义的知名应用占用
    - 注册端口：1024~49151，分配给用户进程或者某些应用程序（TomCat用8080，MySQL用3306）
    - 动态端口：49152~65535，不固定分配给某一进程
  - 协议：数据在网络中传输的规则，常见的协议有UDP协议和TCP协议
    - TCP/IP协议
    - TCP（Transmission Control Protocol）：传输控制协议。面向连接的可靠通信、“三次握手”、“四次挥手”
    - UDP（User Datagram Protocol）：用户数据报协议。　　无连接的不可靠传输协议
- UDP通信
  - DatagramPacket：数据包对象
  - DatagramSocket：发送端和接收端对象

- 广播：广播需要使用广播地址：255.255.255.255。 指定端口：9999

- 组播：需要使用组播地址：224.0.0.0~239.255.255.255 指定端口：9999

- TCP通信
  - 在Java中只要使用了java.net.Socket类实现了通信，底层即是使用了TCP协议
  - Socket类、OutputStream getOutPutStream（）获得输出流对象、InputStream getInputStream（）获得字节输入流对象
  - 线程池优化：ExecutorService
- 即时通信
  - 即时通信是客户端发给别的客户端。过程：客户端->服务器->客户端。（端口转发思想）
- 模拟BS系统

- Java高级技术：单元测试、反射、注释、动态代理
  - 单元测试是针对最小的功能单元编写测试代码，Java中最小的功能单元是方法。
  - Junit单元测试框架（必学）
    - JUnit可以灵活的选择执行哪些测试方法，可以一键执行全部测试方法。可以生成测试报告，某个方法测试失败不会影响其他测试方法功能。
    - 导入jar包，编写测试方法：必须是公共的无参数无返回值的非静态方法，写@Test注解，完成被测试方法的预期正确性测试
    - 常用注解：@Test、@Before、@After、@BeforeClass、@AfterClass
  - 反射是指对于任何一个Class类，在"运行的时候"都可以直接得到这个类全部成分
    - 核心和关键：得到编译以后的class文件对象
    - 反射第一步：获取Class类的对象。三种方法：1、Class.forName（）2、类名.class 3、对象.getClass（）
    - 反射获取Constructor、反射获取成员变量对象和方法对象
    - 反射作用：绕过编译阶段为集合添加数据、通用框架的底层原理、破坏泛型的约束性、破坏封装行、做Java高级框架
  - 注解对Java中的类、方法、成员变量做标记，然后进行特殊处理
    - 自定义注解：public @interface 注解名称 {public 属性类型 属性名（）default 默认值；}
    - 元注解：注解注解的注解，有两个，@Target（约束自定义注解只能在那些地方使用）和@Retention（申明注解的生命周期）

- 动态代理（重难点）：Java中代理的代表类java.lang.reflect.Proxy，类中提供一个静态方法，对对象产生一个代理对象返回

- XML、解析、设计模式
  - XML（eXtensible Markup Language），是一种数据表示格式，可以描述非常复杂的数据结构，常用于传输和存储数据
    - 一种纯文本，UTF-8编码。一般被当成消息进行网络传输，或作为配置文件存储系统信息
    - 跟标签只能有一个、文档生命必须在第一行、标签必须成对出现且正确嵌套
    - <!-- 注释 -->
    - XML文档约束-schema
  - XML解析就是使用程序读取XML数据：JAXP、JDOM、dom4j（常用）、jsoup
    - dom4j解析XML文件、dom4j解析文件中的各种节点
    - Xpath技术

JavaWeb

- MySql
  - net start mysql 启动mysql服务　　net stop mysql 停止mysql服务
  - 用户名：root 密码：123456
  - >mysql -uroot -p123456 -h127.0.0.1 -P3306
  - mysql可以创建多个数据库，每个数据库可以创建多个数据表
  - SQL语句可以单行和多行书写，以分号结尾。语句不区分大小写，关键字建议大写。单行注释：-- 空格 或者 #　　多行注释：/**/
  - SQL分类
    - DDL（Data Definition Language）数据定义语言，用来定义数据库对象：数据库，表，列等
      - 查询：Show DataBases
      - 创建数据库：Create DataBase 数据库名称；　　判断不存在：Create DataBase IF NOT EXISTS
      - 删除数据库： Drop DataBase 数据库名称（判断是否存在同上）
      - 使用数据库：Use DataBase（）
      - 查看当前使用的数据库：SELECT DATABASE（）
      - DDL操作表
        - 查询表结构：DESC 表名称；

- 显示表：SHOW TABLES

- create table 表明名（

  字段名1 数据类型，

  字段名2 数据类型，

  字段名3 数据类型，

  ……

  字段名n 数据类型

  ）ENGINE=INNODB DEFAULT CHARSET=utf8；
- 删除表：DROP TABLE 表名
- 修改表名：ALTER TABLE 表名 RENAME TO 新表名
- 添加一列：ALTER TABLE 表明 ADD 列名 数据类型
- 修改数据类型：ALTER TABLE 表明 MODIFY 列名 新数据类型
- 修改列名和数据类型：ALTER TABLE 表名 CHANGE 列名 新列名 新数据类型
- 删除列：ALTER TABLE 表名 DROP 列名

- DML（Data Manipulation Language）数据操作语言，用来对数据库中表的数据进行增删改
  - 添加数据
    - 给指定列添加数据： INSERT INTO 表名（列名1,列名2，列名3,...）VALUES（值1,值2,值3,...）；
    - 给全部列添加：INSERT INTO 表名 VALUES（值1,值2,值3,...）；
    - 批量添加
  - 修改数据： UPDATE表名 SET 列名1=值1，列名2=值2，...[WHERE 条件]
  - 删除：DELETE FROM 表名 [WHERE 条件]
- DQL（Data Query Language）数据查询语言，用来查询数据库中表的记录

- SELECT
  字段列表
  FROM
  表名列表
  WHERE
  条件列表
  GROUP BY
  分组列表
  HAVING
  分组后条件
  ORDER BY
  排序字段
  LIMIT
  分页字段
- 基础查询
  - 查询多个字段：SELECT 字段列表 FROM 表名；*查询所有数据
  - 去除重复记录：SELECT DISTINCT 字段列表 FROM 表名；
  - AS取别名 as可以省略
- 条件查询：SELECT 字段列表 FROM 表名 WHERE 条件列表
  - &&和and、 ||和or、！和not
  - 模糊查询：like（_单个任意字符，%多个任意字符）,between and,in,is null,<=>(安全等于)
- 排序查询：SELECT 字段列表 FROM 表名 ORDER BY 排序字段名1[排序方式1]，排序字段名2[排序方式2]...；
  - ASC升序排序（默认）、DESC降序排序
  - 如果有多个排序条件，当前边条件值一样时，才会根据第二条件进行排序
- 聚合查询：SELECT 聚合函数名（列名） FROM 表
  - 聚合函数：count（）、max（）、min（）、sum（）、avg（）
- 分组查询：SELECT 字段列表 FROM 表名列表
  [WHERE 分组前条件限定] GROUP BY 分组字段名 [HAVING 分组后条件过滤]
  - 执行顺序：where >聚合函数>having
- 分页查询：SELECT 字段列表 FROM 表名 LIMIT 起始索引，查询条目数；（从0开始）

- DCL（Data Control Language）数据控制语言，用来定义数据库的访问权限和安全级别，及创建用户

- 约束
  - 约束作用于表中列上的规则，用于限制加入表的数据
  -
  - 1、NOT NULL：非空约束，用于约束该字段的值不能为空。比如姓名、学号等。
  - 2、DEFAULT：默认值约束，用于约束该字段有默认值，约束当数据表中某个字段不输入值时，自动为其添加一个已经设置好的值。比如性别。
  - 3、PRIMARY KEY：主键约束，用于约束该字段的值具有唯一性，至多有一个，可以没有，并且非空。比如学号、员工编号等。
  - 4、UNIQUE：唯一约束，用于约束该字段的值具有唯一性，可以有多个，可以没有，可以为空。比如座位号。
  - 5、CHECK：检查约束，用来检查数据表中，字段值是否有效。比如年龄、性别。
  - 6、FOREIGN KEY：外键约束，外键约束经常和主键约束一起使用，用来确保数据的一致性，用于限制两个表的关系，用于保证该字段的值必须来自于主表的关联列的值。在从表添加外键约束，用于引用主表中某列的值。比如学生表的专业编号，员工表的部门编号，员工表的工种编号。
    - 外键用来让两个表得数据之间建立链接，保证数据的一致性和完整性

- 数据库设计：建立数据库中表结构以及表与表之间得关联关系的过程
  - 设计过程：需求分析、逻辑分析（ER图）、物理设计（逻辑到物理转换）、维护设计（新需求、表优化）
  - 多对多可以建立中间表

- 多表查询
  - 内连接查询语法：
    隐式neilianjie
    SELECT 字段列表 FROM 表1，表2... WHERE 条件
    显示内连接
    SELECT 字段列表 FROM 表1 [INNER] JOIN 表2 ON 条件;
  - 外连接查询语法：
    左外链接（左表的所有数据和交集）
    SELECT 字段列表 FROM 表1 LEFT [OUTER] JOIN 表2 ON 条件
    右外链接（右表的所有部分和交集）：
    SELECT 字段列表 FROM 表1 RIGHT [OUTER] JOIN 表2 ON 条件

- 子查询：
  单行单列：
  SELECT 字段列表 FROM 表 字段名 =（子查询）
  多行单列：
  SELECT 字段列表 FROM 表 WHERE 字段名 =（子查询）
  多行多列：
  SELECT 字段列表 FROM （子查询）WHERE 条件

- 事务
  - 包含了一组数据库操作命令，这些 命令要么同时成功，要么同时失败。
  - 开始事务：BEGIN
    回滚事务：ROLLBACK
    提交事务：commit
  - 事务四大特征A（原子性）、C（一致性）、I（隔离性）、D（持久性）

- JDBC
  - Connection 数据库连接对象（事务定义！）：
    开启事务：setAutoCommit（boolean autoCommit）
    提交事务：commit（）
    回滚事务：rollback（）
  - Statement：
    int executeUpdate（sql）执行DML、DDL语句
    ResultSet executeQuery（sql）执行DQL语句
  - ResultSet 结果集对象：
    封装了DQL查询语句的结果
    Boolean next（）
    xxx getXxx（）
  - SQL注入：
    密码写成 '' or '1' = '1'
    - PreparedStatement（解决sql注入）：
      1.获取PreparedStatement对象
      2.设置参数值
      3.执行SQL语句
    - PreparedStatement好处(使用要配置)：
      1.预编译SQL，性能更高
      2.防止SQL注入：将敏感字符进行转义

- 数据库连接池：负责分配、管理数据库连接
  （1、资源复用 2、提升系统响应速度 3、避免数据库连接遗漏）

- Druid连接池是阿里巴巴开源的连接池：导入jar包

- Maven专门用于管理和构建Java项目的工具：
  1.标准化项目结构
  2.标准化构建流程（编译、测试、打包、发布..）
  3.提供一套依赖管理机制（jar包、插件...）
  - 本地仓库、中央仓库、私服仓库
  - 常用命令：complie、clean、test、package、install
  - maven坐标：资源的唯一标识
    1.groupId：maven项目隶属组织名称
    2.artifactId：maven项目名称
    3.version：当亲项目版本号
  - 通过设置坐标依赖范围（scope）
    compile、test、provided、runtime、system、import
- （JavaEE三层架构：表现层、业务层、持久层，持久层是负责保存到数据库的那一层代码）
  - web层：与客户端交互，包括获取用户请求，传递数据，封装数据，展示数据。
    springMVC
  - service层：复杂的业务处理，包括各种实际的逻辑运算。spring
  - dao层：与数据库进行交互，与数据库相关的代码在此处实现。MyBatis
- 框架就是一个半成品软件，是一套可用的、通用的、软件基础代码模型
- MyBatis是一款优秀的持久层框架，用于简化JDBC开发
  - JDBC：1.硬编码　2.操作繁琐
  - 1.加载mybatis的核心配置文件，获取sqlSessionFactory
    2.获取SqlSession对象，用它执行sql
    3.执行sql
    4.释放资源
  - Mapper代理开发：解决原生方式中的硬编码、简化后期执行SQL。
    - 注意在resources下创建xml文件包名和Java下边一致，在Java目录下用.做分隔符、在resources下边用/做分隔符
    - SQL映射文件的namespace属性为Mapper接口的全限定名
    - 如果Mapper接口名称和SQL映射文件名称相同，并在同一目录下，则可以使用包 扫描的方式简化SQL映射文件的加载
  - mybatis配置文件
    - environemnts：配置数据库连接环境信息。可以配置多个environment，通过

default属性　　切换不同的environment

- 配置标签的时候，需要遵循前后顺序

- sql片段（不灵活）
  使用resultMap替换
- 实体类属性名和数据库列名不一致，不能自动封装数据：1、起别名　2、resultMap标签
- 特殊字符用转义字符处理
  #{}执行sql时，会将#{}替换为？，将来自动设置参数值
  ${}会存在SQL注入的问题
- 查询：查询详情、条件查询、动态条件查询（if、where.）
- 添加：添加、主键返回
- 修改：修改全部字段、修改动态字段
- 删除：删除一个、批量删除（foreach标签）
- MyBatis会将一个数组阐述，封装为一个Map集合。*默认：array = 数组
  *使用@Param注解改变map集合的默认key的名称
- 使用注解开发会比配置文件开发更加方便：1、注解完成简单　2、配置文件完成复杂功能

html（结构）+css（表现）+js（行为）

html是一种用于创建网页的标准标记语言。各种标签混合在一起。

- 表单是网页中主要负责数据采集功能，使用<form>标签定义。
  action：规定提交表单时向何处发送表单数据，URL
  method：规定用于发送表单数据的方式
  get：
  post：
- 表单标签-表单项

- css三种导入方式：
  1、内联样式
  2、内部样式
  3、外部样式

js跨平台、面向对象。能够改变图像src属性值、能够进行表单验证

- 输出语句：
  window.alert（）写入警告框
  document.write（）写入html输出
  console.log（）写入浏览器控制台
- var定义变量相当于全局变量、可以重复定义
  let定义变量

- ==：先进行类型转换，再比值
  ===：不会进行类型转换

- 对象
  - JavaScript 对象
    - array数组对象，相当于Java中的集合
    - string对象，
    - 自定义对象
  - browser对象
    - window、navigator、screen、history、location
    - window对象：浏览器窗口对象，window.可以省略。alert（）、confirm（）、setInterval（）、setTimeout（）
  - DOM 对象
    - document对象、element对象、attribute对象、text文本对象、comment对象
    - 改变html元素内容、元素的样式（css）、对DOM事件做出反应、添加和删除html元素
    - 获取element
  - js事件监听
    - 事件绑定：
      1、通过html标签
      2、通过DOM元素属性

- web核心
  - 静态资源：html、css、js等
    动态资源：servlet、jsp
    数据库：负责存储数据
    web服务器：tomcat
  - HTTP、Tomcat、Servlet
    - HTTP基于TCP协议、基于请求-响应模型的，无状态协议、每次请求-响应都是独立的。
      请求格式：1.请求行　2.请求头　3.请求体
      GET请求和POST请求
      响应格式：1.响应行　2.响应头　3.响应体（状态码分类）
    - web服务器：1.封装HTTP协议，简化开发　2.将web项目部署到服务器中，对外提供网上浏览服务
      Tomcat：web容器、servlet容器。servlet需要依赖于Tomcat运行

Tomcat配置maven插件

- Servlet是Java提供的一门动态web资源开发技术。
  - 1、导入servlet依赖坐标 2、定义一个类实现Servlet接口 3、配置，使用@WebServlet注解 4、访问
  - 执行流程：Servlet由web服务器创建，Servlet方法由web服务器调用。
  - 生命周期：1、加载和实例化 2、初始化 3、请求处理 4、服务终止
  - Servlet方法：自定义servlet，会继承HttpServlet。根据请求方式的不同，进行分别的处理
  - urlPattern配置规则：1.精确匹配 2.目录匹配 3.扩展名匹配 4.任意匹配
  - xml配置servlet

- Request（请求）、Response（响应）
  - request获取请求数据、response设置响应数据
  - tomcat需要解析请求数据，封装为request对象，并且创建request对象传递到service方法中
  - 中文乱码：
    1.POST request.setCharacterEncoding（"UTF-8"）
    2.GET tomcat默认ISO-8859-1解码。先对乱码数据进行编码,转为字节数组（String.getBytes（StandardCharsets.ISO_8859_1）），然后字节数组解码(new String (bytes, StandardCharsets.UTF_8))
  - URL编码实现方式：URLEncoder.encode（str，“utf-8”）
    解码：URLEncoder.decode（str，“utf-8”）
  - 请求转发forward：
    req.getRequestDispater（“”）.forward（req， resp）
    - 地址栏路径不发生变化
    - 只能转发到当前服务器的内部资源
    - 一次请求，可以转发的资源间使用request共享数据
  - response重定向（redirect）：
    - 浏览器地址发生改变
    - 可以重定向到任意资源位置
    - 两次请求，不能再多个资源使用request共享数据
  - response响应字符数据流、字节数据流
  - SqlSessionFactory工厂只创建一次，使用代码块：

- JSP、会话技术（Cookie、Session）

- Servlet --> JSP --> Servlet+JSP --> Servlet+html+ajax

- EL表达式
  - JavaWeb四大域对象：page、request、session、application
  - EL表达式获取数据，会议依次从这四个域中寻找
- MVC+三层结构的开发思想
- 创建Cookie对象：Cookie cookie = new Cookie（“key”,“value”）
  - 基于HTTP协议，响应头、请求头
  - 可以设置存活时间
  - cookie不能直接存储中文
- session：服务器会话跟踪技术，将数据保存到服务器
  - Session钝化、活化。session销毁

- Filter（过滤器）、Listener（监听器）
  - 过滤器可以把对面资源拦截下来，实现一些特殊的处理。（拦截具体资源、目录拦截、后缀名拦截、拦截所有）
  - 1.定义类，实现Filter接口
    2.注解@WebFilter配置拦截路径
    3.在doFilter方法中放行
  - 配置多个过滤器，优先级按照类名的自然排序执行
  - 监听器可以监听就是在application、session、request三个对象创建、销毁或者往其中添加修改删除属性时自动执行代码的功能组件
- Ajax、Vue、Element UI
  - Ajax-异步的js和xml。作用：与服务器进行数据交换。
  - 步骤：1.编写AjaxServlet，并使用response输出字符串　2.创建XMLHttpRequest对象：用于和服务器交换数据　3.向服务器发送请求　4.获取服务器响应数据
  - AXIOS框架
  - JSON
  - VUE免除原生js中DOM操作，简化书写。常用指令：v-bind、v-model、v-on、v-for
  - Element是一套基于Vue的网站组件库，用于快速构建网页。
- JavaWeb三大组件：Servlet、Filter、Listener

Java开发框架

- **Java开发框架**
  - Maven
    - 项目管理工具。核心：pom.xml
    - maven坐标，使用唯一标识，唯一性定位资源位置
    - mvn compile
      mvn clean
      mvn test
      mvn package
      mvn install
    - 1、maven创建不同项目。2、web工程打包为war、java工程打包为jar。
    - 依赖范围：依赖的jar可以在任何地方使用，通过scope标签设置其作用范围
    - 生命周期：complie、test-complie、test、package、install
      maven对象项目构建分为三个阶段：clean、default、site
    - maven高级
      - 分模块开发与设计：控制层、数据层、业务层、服务层。接口进行通信
      - 聚合：多模块构建维护，<packaging>、<modules>
      - 继承：模块依赖关系维护。<dependencyManagement>
      - 聚合用于快速构建项目，继承用于快速配置
      - 属性：定义自定义属性。
      - 版本管理：RELEASE、SNAPSHOT
      - 资源配置：
      - 环境配置：多环境开发配置
      - 跳过测试
      - 私服
        - nexus是一个maven私服产品。
        - 宿主仓库、代理仓库、仓库组
  - Git
    - 备份、代码还原、协同开发、追溯问题代码的编写人和编写时间。Git分布式版本控制工具
    - git add （工作区-->暂存区）
      git commit （暂存区-->本地仓库）
    - git reset --hard commitID
  - SSM

- SSM
  - spring
    - 简化开发（IoC、AOP）、框架整合。主要学习Spring Framework 、Spring Boot 、 Spring Cloud
    - Spring Framework是Spring生态圈中最基础的项目。
      - Core Container：核心容器
        - IoC/DI
          - Ioc：控制反转，在程序中不适用new创建对象，转换为外部提供对象。（为了解耦）
          - spring提供了一个容器，称为IoC容器，IoC容器扶着对象创建、初始化工作
          - DI：依赖注入。在容器中建立bean与bean之间的依赖关系的整个过程，称为依赖注入。
            1.删除使用new的形式创建对象 2.提供依赖对象对应的setter方法
            3.配置service和dao之间的关系
        - IoC容器：管理service和dao
        - 依赖注入方式：setter注入，构造器注入，集合注入
          - 强制依赖使用构造器进行、自己开发的模块推荐使用setter注入
          - 自动装配：IoC根据bean所依赖的资源在容器中自动查找并注入到bean中的过程称为自动装配
          - 加载properties文件：1.开context空间 2.使用context空间加载properties文件 3.使用属性占位符${}读取properties文件中的属性
        - Bean：bean默认为单例对象（通过scope属性更改）
          - 1.构造方法实例化bean 2.使用静态工厂实例化bean 3.使用实例工厂实例化bean 4.使用FactoryBean实例化工厂
          - bean生命周期控制和关闭容器
        - BeanFactory是IoC容器的顶层接口，ApplicationContext是Spring容器的核心接口
      - Aspects：AOP思想实现
      - AOP：面向切面编程，指导开发者如何组织程序结构。作用：在不惊动原始设计的基础上为其进行功能增强
        - 连接点、切入点、通知、通知类、切面
          通配符、语法格式
        - 开发模式：xml、注解

- @Compent告诉spring、@Aspect做AOP的、@Pointcut切入点 、@Befor提前执行、@EnableAspectJAutoProxy开启spring对aop注解驱动支持
- Data Access：数据访问
- Data Integration：数据集成
- Web：Web开发
- Test：单元测试与集成测试

- 注解开发
  - @Configuration注解用于设定当前类为配置类
    @ComponetScan注解用于设定扫描路径，如果多个可以用{}。
    @Componen用@Controller、@Service、@Repository代替
  - @Scope控制范围
  - 自动装配：@Autowired（不推荐）、@Qualifier、@Value（简单类型、可以加载properties文件数据）
  - 管理第三方bean：1.定义一个方法获取要管理的对象。 2.添加@Bean 3.或者使用@Import
- Spring整合Mybatis、JUnit
  - 添加对应的pom.xml坐标
    整合JUnit @Runwith、@ContextConfiguration
- Spring事务：在数据层和业务层都可以使用。PlatformTransactionManager接口

- springMVC：web层开发技术
  - SpringMVC简介
    - SpringMVC是轻量级Web框架
    - @EnableWebmvc
    - 日期类型传输
  - 请求与响应
    - 文本数据
    - json数据
  - REST风格（访问网络资源的格式）
    - 约定
      GET：查询用户信息
      POST：新增/保存信息
      PUT：修改/更新信息
      DELETE：删除

- DELETE：删除
- SSM整合
    - 表现层数据封装
    - 异常处理器：@ExceptionHandler
        - 业务异常
        - 系统异常
        - 其他异常
    - 拦截器简介
        - 拦截器是一种动态拦截方法调用的机制
    - 项目异常方案
- 拦截器
- springBoot
    - 简化spring应用的初始搭建以及开发过程
    - Spring缺点：1.配置繁琐 2.依赖设置繁琐
      SpringBoot优点：1.自动配置 2.起步依赖（简化依赖配置） 3.辅助功能（内置服务器，...）
    - 使用@RestController @RequestMapping注解
    - 使用maven package打成jar包，在目录target/test-classes使用命令java -jar 文件名 运行
    - Jetty服务器比Tomcat更加轻量，扩展性更强
    - 修改端口号
        - 1.application.properties server.port=
        - 2.application.yml server： port：（80前面一定有一个空格）
        - 3.application.yaml server： port：
    - yml格式和yaml格式：千万不要忘记空格！！！
        - 读取方式一：@Value
        - 读取方式二：Environment类getProperty方法
        - 读取方式三：实体类封装属性 @ConfigurationProperties
        - yml多环境开发：---
    - maven为主，SpringBoot为辅
    - SpringBoot的SSM整合案例

        - pom.xml配置起步依赖，必要的资源坐标
        - application.yml设置数据源、端口
        - 配置类：全部删除
        - dao：设置@Mapper
    - MyBatisPlus
        - 基于MyBatis框架基础上开发的增强型工具，旨在简化开发、提高效率
        - SpringBoot整合MyBatisPlus
        - 无侵入：只做增强不做改变，引入它不会对现有工程产生影响，如丝般顺滑
        - 强大的 CRUD 操作：内置通用 Mapper、通用 Service，仅仅通过少量配置即可实现单表大部分 CRUD 操作，更有强大的条件构造器，满足各类使用需求
        - 支持 Lambda 形式调用：通过 Lambda 表达式，方便的编写各类查询条件，无需再担心字段写错
        - 乐观锁
        - 代码生成器
    - 深入Spring
    - SpringBoot
    - MybatisPlus
- **中间件&服务框架**
- **项目实践**
- **面试专题**