两层模式C/S，三层体系结构：用户界面表示层、业务逻辑层、数据访问层

企业级应用特点：灵活性、可扩展性、可共享性、安全性、重复可用性、稳定性、执行效率、容错能力、负载平衡能力

javaee一种体系结构和一套标准，有多种技术和框架，为了简化开发提高项目效率

编程思想：组件(有独立功能的类)+容器(组件运行的环境)=提供底层服务, 惯例优于配置，配置优于编码

分层模型(松耦合，轻量级)：对象层(POJO)数据访问层(DAO)业务逻辑层(Service)控制层(Controller)表现层(View)

优点：1独立于硬件配置和操作系统2坚持面向对象的设计原则3灵活性，可移植性，互操作性4轻松的企业信息系统集成

Servlet：运行在服务端的小程序，本质是类(接口和实现类)，源是.java文件，是一种Javaee的组件技术

动作：接受请求(request)，给出响应(response)

生存周期方法：init()初始化配置(只调用一次)，service()响应客户请求，destroy()销毁servlet对象回收资源(关闭服务器时调用)

非生命周期方法：getServletInfo()、getServletConfig()

运行阶段：每一个servlet创建一个对象，每一次浏览器的请求，服务器产生一个线程，调用service()，线程反复执行service()

HttpServlet类两个核心方法：doGet(request, response)，doPost(request, response)

HttpServlet类两大功能：1自定义处理方法2调用service方法，处理HTTP请求和响应

每个servlet都有一个上下文实例——ServletContext，一个配置——servletConfig

Filter(过滤器)：拦截请求响应对象

生命周期：1web.xml实例化2.init()初始化3.doFilter()过滤4.destroy()销毁

DoFilter(request, response)方法中chain.doFilter(request, response)用于继续转发请求，转向下一个过滤器或servlet，若未调用这一方法，请求无法进行转发响应，可用来过滤和控制

适用场合：对用户请求进行统一认证(统一编码)，日志审核和记录，图像转换，数据压缩，加密/解密，身份验证，资源访问触发事件过滤

Listener(监听器)：监听另一个java对象的变化——方法监听，属性监听，事件监听

常见三种：上下文监听context、会话监听session、请求监听request

jsp动作：

1include动作<jsp:include page=”路径” flush=”true”/>资源插入引用

2Forward动作<jsp:forward page=”path”}/>请求转发

3useBean动作<jsp:useBean id=”name” class=”className” scope=”作用域对象”/>

4getProperty动作<jsp:getProperty>获取javaBean属性值

5setProperty动作<jsp:setProperty>设置javaBean属性值

6param动作<jsp:param name=”result” value=”110”/>键值对形式

7plugin动作

内置作用域对象：page当前页面request一个请求过程session一个会话application服务器运行过程中

JSP九大内置对象(隐含对象) 无需预先声明/new获取实例就可以直接使用

输入输出：out向客户端浏览器输出数据、response封装服务器响应信息、request封装来自客户端浏览器的信息

通信控制：pageContext提供对jsp页面所有对象及命名空间的访问、session保存会话信息可在同一用户不同请求之间共享、application代表当前应用程序的上下文可在不同的用户之间共享

Servlet：page指向当前jsp程序本身、config封装应用程序配置信息

错误处理：exception封装jsp程序执行过程中发生的异常和错误信息

服务器应用对象：

ServletRequest向服务端传递参数

ServletResponse回应客户端请求

ServletConfig获取servlet初始化参数

ServletContext存储整个web应用相关信息，存储全局共享数据，在一个Web服务器中唯一，所有servlet共享共享同一个ServletContext对象

获取方式：1ServletConfig对象中获取、2HttpServlet对象中获取

应用：

1实现数据共享(数据在服务器运行过程中有效)

2获取Web应用(上下文)初始化参数(web.xml)

3获取服务器相关信息

4读取资源文件(本地资源路径、指定参数的文件等)

5记录日志(向servlet日志文件写入错误日志和异常等)

6HttpSession由服务器创建在服务端存放用户会话状态和信息

方法：

1setAttribute(String name, Obiect object)将对象绑定到会话中使之成为会话属性

2getAttribute(String name)返回会话属性

3getAttributeNames()返回所有会话属性名

4removeAttribute(name)删除会话属性

5invalidate()会话中止并删除绑定的所有会话属性(关闭浏览器时调用)

6setMaxInactiveInterval(int interval)设置会话的超时时间(负永不过期0立即失效)

Ajax(异步的JavaScript和XML)：AB执行功能，A执行不会影响B执行，AB可同时执行。异步过程调用后，调用者无需得到结果便可继续执行后续操作，其调用返回也不受调用者控制

网页的异步更新，局部刷新：

1向服务器发请求的js对象：XMLHttpRequest(XHR)，使用Ajax就是使用XHR对象的属性方法和事件

2谁发请求谁接响应：js脚本中对象向服务器发送请求，对象接受响应，通过js脚本刷新局部页面

3html，js的编写修改不需重启服务器，servlet，配置，JavaBean的编写修改需要重启服务器

EL表达式：存取数据的语言，计算和输出java对象

1直接访问JavaBean中的属性：${user.userName} 或者 ${user[userName]}

2可执行运算：逻辑运算，算术运算，条件运算，关系运算等

3可获得命名空间

4除0无异常（返回无穷大Infinity）

5可访问作用域对象(page,request,session,application)

6输出文本：${“文本”}

JSTL：取代在jsp中嵌入java代码的做法，提高可维护性

核心标签库：表达式操作<c:out><c:set><c:remove><c:catch>流程控制<c:if><c:choose><c:when><c:otherwise>迭代操作：<c:forEach><c:forTokens>

MVC分层：Model模型用来处理业务，View视图用于显示，Controller控制器用来控制分派

实现功能：1jsp做界面2servlet处理请求3service处理业务(接口和实现)4dao处理数据库访问和操作5pojo(entity)封装数据6junit做单元测试7DBUtils封装数据库访问工具和方法

项目分层结构：表示层→控制层→业务层→DAO层→数据访问层，依次依赖

数据传递：jsp发出请求传递参数→servlet接受请求调用service→service调用dao层→dao访问数据库处理业务

，调用pojo封装数据返回结果→service得到数据→servlet响应发送数据→jsp显示数据

控制反转依赖注入

IOC在使用Spring框架之后，对象的实例不再由调用者来创建，而是由Spring容器来创建，Spring容器会负责控制程序之间的关系，而不是由调用者的程序代码直接控制。控制权由应用代码转移到了Spring容器，控制权发生了反转，即为控制反转。

DI从Spring的角度去看，Spring容器负责将被依赖对象赋值给调用者的成员变量，相当于为调用者注入了它依赖的实例，即为依赖注入。

javaBean的构造器（构造方法）访问器（访问方法）和设置器（设置方法）

JavaBean组件就是java代码封装的类

Javabean规范

1公共的无参构造方法：一个javaBean对象必须拥有一个公共类型和默认的无参构造方法，从而可以通过new关键字直接对其实例化

2类的声明是非final类型：当一个类声明为final类型时，它是不可以更改的，所以javaBean对象声明应该是非final类型

3实现可序列接口：javaBean应该直接或间接实现java.io.Serializable接口，以支持序列化机制

4为属性声明访问器：javaBean中的属性应该设置为private私有类型，为了防止外部直接访问，需要对外提供公共public的访问方法，即需要为属性提供getter/setter方法

JDBC：规避数据库的不同，为开发人员访问数据库提供统一的编程接口API

JDBCAPI做三件事：创建连接，创建执行语句，执行语句返回结果

DriverManager：依赖数据库不同，管理加载驱动

Connection：连接数据库和传送数据

Statement：由Connection产生，负责执行sql语句

ResultSet：保存Statement执行后产生的处理结果

1加载驱动

Class类的静态方法forName来Class.forName(jdbcDriver)

或MySql-Class.forName(“com.mysql.jdbc.Driver“）

2创建连接

DriverManager获取Connection conn=DriverManager.getConnection(url,username,password)

或MySql-url="jdbc:mysql://localhost:3306/database"

3准备sql语句：通过Connection对象创建Statemenbt对象

①String sql="....."

②PreparedStatement ps = conn.prepareStatement(sql)预编译对象

或Statement ps=conn.createStatement(sql)基本对象

4执行sql语句：使用Statement对象

ResultSet rs=ps.executeQuery()查询语句

int num=ps.executeUpDate()增删改语句

ResultSet rs=ps.execute()通用执行任何SQL语句

5.处理结果：返回ResultSet对象来取出查询结果

while(rs.next()){

String xx=rs.getString("xx");.......}

6回收资源：关闭ResultSet，关闭Statement，关闭Connection

rs.close(); Ps.close(); Conn.close();

Servlet调用过程：

用户点击一个链接，链接的URL指向一个servlet

容器“看出”这个请求指向某个Servlet，所以容器创建两个对象 HttpServletRequest 。HttpServletResponse

容器根据请求中的URl找到正确的servlet，为这个请求创建或分配一个线程，并调用servlet的service()方法，把请求和响应对象作为参数传递给它

service()方法根据客户发出的HttP方法（GET、POST），确定要调用那个servlet方法。

客户端发出了一个HTTP get请求，所以service（）方法会调用servle的doGetI()方法，把请求和相应对象作为参数传给它

servlet使用相应对象将相应写个客户，响应通过容器传回。

service()方法结束，所以线程要么撤销，要么返回到容器管理的线程池。请求和响应对象引用超出作用域，可以进行垃圾回收

客户得到响应。

JSP：含有java写成的脚本的动态页面

注释：<%--注--%>客户端不能看<!--注--> 客户端能看

变量声明：<%!%>，jsp表达式：<%=%>，java代码块：<%%>

Jsp指令：<%@指令名属性=”值”%>

page指令：<%@page import=”java.util.\*”%>定义jsp页面全局属性

include指令：用于包含其他jsp、html、文本文件，被包含文件被加载在指令位置

taglib指令：允许用户使用自定义标签

绝对路径action=”http://localhost:8080/login/test”相对路径action=”test”，相对路径简洁方便，相对当前文件所在，只能通过内部访问，两者都能成功访问对应servlet，服务器会自动分析截取url，然后与web.xml文件中的url匹配