JavaWeb 期末开卷知识点清单

90 分钟开卷速查版 | JavaBean + JSP + Servlet、EL、JSTL、Filter、上传下载、JDBC、C3P0、DBUtils

资料来源：老师划定重点；你的实验二至实验八；教材 PPT 第 4、6、7、8、9、10、11 章，以及上机作业 PPT。已额外渲染检查 PPT 和 Word 页面中的截图/流程图/代码图，重点图示内容已转成文字速查项。

目录

说明：本页是静态目录，确保打印后也能直接翻阅；Word/WPS 左侧导航栏还可按标题快速跳转。

0 考场翻阅路线

如果题目出现	先翻	马上写什么
登录、查询、列表、MVC	第 1、10 节	JSP 表单 -> Servlet 控制 -> DAO/DBUtils -> request 域 -> JSP 显示
ServletContext、初始化参数、跳转	第 2 节	getServletContext、getInitParameter、forward/include/sendRedirect
JSP 时间页、异常页、包含	第 3 节	page 指令、errorPage/isErrorPage、jsp:include
${...}、表格显示集合	第 4、5 节	EL 取域对象 + JSTL c:forEach/c:choose
自动登录、权限、编码统一处理	第 6 节	Filter doFilter + Cookie/Session + chain.doFilter
上传、下载、中文文件名	第 7 节	multipart/form-data + @MultipartConfig + Part + Content-Disposition
JDBC 原生操作	第 8 节	Class.forName -> getConnection -> PreparedStatement -> ResultSet -> close
连接池、DBUtils、C3P0	第 9 节	c3p0-config.xml + ComboPooledDataSource + QueryRunner + Handler

答题顺序建议：先画层次或流程，再写核心类/注解/配置，最后补编码、资源释放、中文文件名、异常处理这些得分点。

1 JavaBean + JSP + Servlet 总模型

1.1 JavaBean 规则

- JavaBean 是可复用的 Java 类，用来封装数据或业务组件。实体类必须有 public 无参构造，属性 private，提供 public getter/setter。
- EL 访问 ${u.name} 本质上调用 getName()；DBUtils 的 BeanHandler/BeanListHandler 也依赖字段名与 setter 对应。
- 布尔属性可用 isXxx()；集合、Map 也能作为 Bean 属性被 JSP/EL 展示。

```
public class Student {
    private int id;
    private String name;
    private String snumber;
    private String sex;
```

```
    private int age;
    public Student() {}
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

1.2 JSP 中使用 JavaBean

动作	作用	速记
<jsp:useBean>	创建或查找指定 scope 中的 Bean	id 是变量名，class 是完整类名，scope 可选 page/request/session/application
<jsp:setProperty>	给 Bean 属性赋值	property="*" 可按请求参数名批量注入同名属性
<jsp:getProperty>	输出 Bean 属性	等价于调用 getter 并输出

```
<jsp:useBean id="user" class="com.example.model.User" scope="request"/>
<jsp:setProperty name="user" property="username"/>
<jsp:getProperty name="user" property="username"/>
```

1.3 Model1、Model2 与 MVC

模式	结构	优点	缺点/考点
JSP Model1	JSP 同时负责显示和业务处理	简单、文件少	代码混杂，维护差，适合小例子
JSP Model2	JSP 负责显示，Servlet 负责控制，JavaBean/DAO 负责模型	职责清晰，实验八登录和学生列表就是这个结构	要会 forward/redirect 和域对象传值
MVC	Model 数据/业务，View 页面，Controller 控制器	大型项目常用	题目要求 MVC 时按 controller、dao、model、util、jsp 分包写

```
浏览器 -> login.jsp -> LoginServlet
LoginServlet -> UserDao.login(username, password)
成功: session.setAttribute("loginUser", user); redirect 到 /studentList
StudentListServlet -> StudentDao.getAllStudents()
```

req.setAttribute("studentList", students); forward 到 result.jsp

result.jsp –> JSTL/EL 显示 ${studentList}

2 Servlet 基础

2.1 生命周期与常用类

对象/方法	作用	常考写法
init(ServletConfig)	Servlet 初始化，只执行一次	读取初始化参数、准备资源
service(req, resp)	每次请求执行，分发到 doGet/doPost	一般继承 HttpServlet，重写 doGet/doPost
destroy()	销毁前执行一次	释放资源
HttpServletRequest	请求对象	getParameter、getRequestDispatcher、getSession、getCookies、setAttribute
HttpServletResponse	响应对象	setContentType、getWriter、sendRedirect、addCookie、setHeader
ServletContext	整个 Web 应用共享对象	getServletContext、setAttribute、getAttribute、getRealPath

2.2 注解配置与初始化参数

```
@WebServlet(
    urlPatterns = "/showParams",
    initParams = {
        @WebInitParam(name = "username", value = "admin"),
        @WebInitParam(name = "password", value = "123456")
    }
)
public class InitParamServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        resp.setContentType("text/html;charset=UTF-8");
        String user = getInitParameter("username");
        String pass = getInitParameter("password");
        resp.getWriter().write(user + ":" + pass);
    }
```

```
}
```

2.3 ServletContext 共享数据

```
// 写入，所有 Servlet 可共享
ServletContext context = getServletContext();
context.setAttribute("sharedInfo", "共享数据");

// 读取
String info = (String) getServletContext().getAttribute("sharedInfo");

// 获取 Web 应用内资源真实路径
String path = getServletContext().getRealPath("/WEB-INF/classes/db.properties");
```

2.4 转发、包含、重定向

方式	代码	特点
请求转发 forward	req.getRequestDispatcher("/result.jsp").forward(req, resp);	服务器内部跳转，地址栏不变，共享 request 域
请求包含 include	req.getRequestDispatcher("/a").include(req, resp);	把另一个资源输出合并进当前响应
重定向 sendRedirect	resp.sendRedirect(req.getContextPath()+"/login.jsp");	浏览器重新请求，地址栏变，request 数据丢失，可避免表单重复提交
Refresh 跳转	resp.setHeader("Refresh","3;URL=index.jsp");	实验二用过，延迟跳转

坑点：forward 之前不要已经提交响应；redirect 写路径时优先加 request.getContextPath()，避免项目名变化后路径失效。

3 JSP 技术

3.1 JSP 运行原理

- JSP 第一次访问会被翻译成 Servlet，再编译成 class，然后由容器创建实例并执行。
- 生命周期方法：jspInit() 只初始化一次，_jspService() 每次请求执行，jspDestroy() 销毁前执行。
- 第一次访问慢，后续访问快；JSP 本质仍是 Servlet。

3.2 三类脚本元素

元素	语法	用途	注意
脚本片段	<% Java 代码 %>	写局部变量、流程控制、调用对象	不能在里面声明方法
声明	<%! 成员变量/方法 %>	声明成员变量或方法	成员变量多线程共享，慎用
表达式	<%= 表达式 %>	输出结果到页面	表达式后不能写分号
JSP 注释	<%-- 注释 --%>	服务器端删除	浏览器源代码看不到，比 HTML 注释安全

3.3 page 指令与异常页

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
        import="java.util.*,java.text.SimpleDateFormat"
        errorPage="handleError.jsp" %>

<!-- 错误页必须打开 isErrorPage 才有 exception 隐式对象 -->
<%@ page contentType="text/html;charset=UTF-8" isErrorPage="true" %>
错误类型：<%= exception.getClass().getName() %>
错误消息：<%= exception.getMessage() %>
```

3.4 include 指令与 jsp:include

方式	语法	时机	差别
静态包含	<%@ include file="time.jsp" %>	翻译阶段合并源码	file 不支持表达式；被包含页最好不要有完整 html/body
动态包含	<jsp:include page="time.jsp"/>	请求阶段执行目标资源	page 可为动态文件；实验四显示当前时间用过
转发动作	<jsp:forward page="login.jsp"/>	请求阶段转发	当前页后续内容不再执行

3.5 9 个隐式对象

对象	类型	作用

request	HttpServletRequest	一次请求数据，表单参数，转发共享
response	HttpServletResponse	响应、重定向、响应头
session	HttpSession	一次会话数据，登录用户、购物车
application	ServletContext	整个 Web 应用共享
out	JspWriter	向页面输出，带缓冲
pageContext	PageContext	页面上下文，可取其他 8 个对象，findAttribute 按 page/request/session/application 查找
config	ServletConfig	当前 JSP/Servlet 配置
page	Object	当前 JSP 翻译后的 Servlet 实例
exception	Throwable	错误页中使用

4 EL 表达式

4.1 基本语法与取值顺序

- 语法：${表达式}。主要用来替代 JSP 表达式，输出域对象、请求参数、Bean 属性和集合内容。
- 未指定作用域时，查找顺序为 pageScope -> requestScope -> sessionScope -> applicationScope。
- 访问 JavaBean 属性：${user.username} 调用 getUsername()；访问 Map/List/数组可用点号或中括号。

```
${requestScope.user.username}

${sessionScope.loginUser.username}

${param.username}

${paramValues.hobby[0]}

${header["User-Agent"]}

${cookie.autoLoginCookie.value}

${pageContext.request.contextPath}

${empty userList ? "暂无数据" : "有数据"}
```

4.2 EL 隐式对象

对象	用途
pageScope/requestScope/sessionScope/applicationScope	按指定作用域取属性

param / paramValues	取请求参数单值/多值
header / headerValues	取请求头
cookie	取 Cookie 对象，再用 .value 取值
initParam	取 Web 应用初始化参数
pageContext	取 request、session、servletContext、contextPath 等

4.3 常用运算符

类别	例子	说明
算术	${1+2}	+ - * / %
关系	${age >= 18}	eq/ne/lt/le/gt/ge 也可用
逻辑	${not empty user and user.vip}	and/or/not
empty	${empty list}	null、空字符串、空集合都为 true
三目	${flag ? "是" : "否"}	适合页面轻量判断

5 JSTL

5.1 依赖与 taglib

```
<!-- Jakarta/Tomcat 10+ 实验环境常用 -->
<%@ taglib prefix="c" uri="jakarta.tags.core" %>

<!-- 老教材/旧项目可能出现 -->
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!-- Maven 依赖示例 -->
<dependency>
  <groupId>jakarta.servlet.jsp.jstl</groupId>
  <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
  <version>3.0.0</version>
</dependency>
```

```
<dependency>
  <groupId>org.glassfish.web</groupId>
  <artifactId>jakarta.servlet.jsp.jstl</artifactId>
  <version>3.0.1</version>
</dependency>
```

5.2 核心标签速查

标签	用途	模板
c:out	安全输出	<c:out value="${user.name}" default="匿名"/>
c:set	设置域属性	<c:set var="msg" value="ok" scope="request"/>
c:remove	删除域属性	<c:remove var="msg" scope="request"/>
c:if	单分支判断	<c:if test="${not empty user}">...</c:if>
c:choose/when/otherwise	多分支判断	<c:choose><c:when test="${empty param.x}">...</c:when></c:choose>
c:forEach	遍历集合/数组/Map	<c:forEach items="${users}" var="u" varStatus="s">...</c:forEach>
c:url	生成带上下文路径 URL	<c:url value="/download" var="u"><c:param name="id" value="${id}"/></c:url>
c:redirect	重定向	<c:redirect url="/login.jsp"/>

5.3 表格显示 List 模板

```
<table>
  <tr><th>姓名</th><th>性别</th><th>年龄</th></tr>
  <c:forEach items="${users}" var="u" varStatus="s">
    <tr>
      <td>${u.name}</td>
      <td>${u.sex}</td>
      <td>${u.age}</td>
    </tr>
  </c:forEach>
</table>
```

6 过滤器 Filter

6.1 Filter 工作流程

- Filter 位于客户端和目标资源之间，可拦截 JSP、Servlet、HTML、静态资源。
- 核心方法：init() 初始化，doFilter() 每次拦截执行，destroy() 销毁。
- 必须调用 chain.doFilter(request, response) 才会放行；不调用就是拦截终止。
- Filter 链按映射顺序执行：请求时 A -> B -> 资源，响应返回时 B -> A。

```
@WebFilter("/*")
public class EncodingFilter implements Filter {
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
            throws IOException, ServletException {
        req.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html;charset=UTF-8");
        chain.doFilter(req, resp);
    }
}
```

6.2 自动登录模板

```
@WebFilter("/*")
public class AutoLoginFilter implements Filter {
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,
                         FilterChain chain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) servletRequest;
        HttpServletResponse response = (HttpServletResponse) servletResponse;

        User loginUser = (User) request.getSession().getAttribute("loginUser");
        if (loginUser == null) {
            Cookie cookie = CookieUtil.getCookie(request, "autoLoginCookie");
            if (cookie != null) {
                String[] values = cookie.getValue().split(":");
                if ("admin".equals(values[0]) && "123456".equals(values[1])) {
                    request.getSession().setAttribute("loginUser", new User(values[0], values[1]));
                }
            }
```

```
        }
        chain.doFilter(request, response);
    }
}
```

6.3 Cookie 与 Session 得分点

场景	写法
登录成功保存用户	request.getSession().setAttribute("loginUser", user);
记住我 Cookie	new Cookie("autoLoginCookie", username + ":" + password); setMaxAge(7*24*60*60); setPath("/");
退出登录	request.getSession().invalidate(); Cookie 同名空值 + setMaxAge(0) 删除
只取已有 session	request.getSession(false)，避免未登录时新建 session

7 文件上传与下载

7.1 上传核心

- 表单必须 method="post" 且 enctype="multipart/form-data"。
- Servlet 必须加 @MultipartConfig，使用 request.getPart("file") 获取 Part。
- 保存路径可用 getServletContext().getRealPath("/upload/")，真实项目更建议保存到应用外固定目录。
- 文件名用 part.getSubmittedFileName()，保存前应检查空值、非法路径和重名。

```
<form action="upload" method="post" enctype="multipart/form-data">
  <input type="file" name="file" required>
  <button type="submit">上传文件</button>
</form>
```

```
@WebServlet("/upload")
@MultipartConfig
public class UploadServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String uploadPath = getServletContext().getRealPath("/upload/");
        File folder = new File(uploadPath);
```

```
        if (!folder.exists()) folder.mkdirs();
        Part part = request.getPart("file");
        String fileName = part.getSubmittedFileName();
        part.write(uploadPath + File.separator + fileName);
        response.sendRedirect("index.jsp");
    }
}
```

7.2 下载核心

- 下载本质是读取服务器文件，通过 response 输出字节流。
- 关键响应头：Content-Type 设置二进制，Content-Disposition 设置 attachment 和文件名。
- 中文文件名：URLEncoder.encode(fileName, UTF-8) 后替换 + 为 %20 更稳。

```
@WebServlet("/download")
public class DownloadServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
        String fileName = request.getParameter("filename");
        String uploadPath = getServletContext().getRealPath("/upload/");
        File file = new File(uploadPath, fileName);
        if (!file.exists()) {
            response.sendError(HttpServletResponse.SC_NOT_FOUND, "文件不存在");
            return;
        }
        String encoded = URLEncoder.encode(fileName, StandardCharsets.UTF_8).replace("+", "%20");
        response.setContentType("application/octet-stream");
        response.setHeader("Content-Disposition", "attachment; filename*=UTF-8''" + encoded);
        try (FileInputStream in = new FileInputStream(file);
             OutputStream out = response.getOutputStream()) {
            byte[] buf = new byte[1024];
            int len;
            while ((len = in.read(buf)) != -1) out.write(buf, 0, len);
        }
    }
}
```

8 JDBC

8.1 JDBC 六步

1. 导入数据库驱动 JAR，例如 mysql-connector-j。
2. 加载驱动：Class.forName("com.mysql.cj.jdbc.Driver")。新版 JDBC 可自动加载，但考试写上更稳。
3. 获取连接：DriverManager.getConnection(url, username, password)。
4. 创建 PreparedStatement：conn.prepareStatement(sql)。
5. 设置参数并执行：setXxx，executeQuery 或 executeUpdate。
6. 处理 ResultSet，最后按 ResultSet、Statement、Connection 顺序关闭资源。

```
String url = "jdbc:mysql://localhost:3306/test?useSSL=false&serverTimezone=UTC&characterEncoding=utf8";
Class.forName("com.mysql.cj.jdbc.Driver");
try (Connection conn = DriverManager.getConnection(url, "root", "123456");
     PreparedStatement ps = conn.prepareStatement("select * from user where username=? and password=?")) {
    ps.setString(1, username);
    ps.setString(2, password);
    try (ResultSet rs = ps.executeQuery()) {
        if (rs.next()) {
            User user = new User();
            user.setId(rs.getInt("id"));
            user.setUsername(rs.getString("username"));
        }
    }
}
```

8.2 API 速查

API	作用	常考点
DriverManager	加载驱动、创建连接	getConnection(url,user,pwd)
Connection	代表数据库连接	prepareStatement、createStatement、close
Statement	执行静态 SQL	易 SQL 注入，不推荐拼接用户输入
PreparedStatement	预编译 SQL	用 ? 占位，setString/setInt 防注入
ResultSet	查询结果集	next() 移动游标；列索引从 1 开始；getXxx("列名")

executeQuery	执行 SELECT	返回 ResultSet
executeUpdate	执行 INSERT/UPDATE/DELETE/DDL	返回影响行数，DDL 常返回 0
execute	执行任意 SQL	返回 boolean，少用

8.3 原生 DAO 模板

```
public User login(String username, String password) {
    String sql = "select id, username, password from user where username=? and password=?";
    try (Connection conn = JDBCUtils.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setString(1, username);
        ps.setString(2, password);
        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                User u = new User();
                u.setId(rs.getInt("id"));
                u.setUsername(rs.getString("username"));
                u.setPassword(rs.getString("password"));
                return u;
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}
```

9 C3P0 与 DBUtils

9.1 数据库连接池

- 连接池预先创建并管理若干 Connection，应用程序用完归还，不频繁创建/关闭物理连接。
- DataSource 是 JDBC 标准数据源接口，核心方法 getConnection()。
- C3P0 的核心类是 ComboPooledDataSource，可读取 c3p0-config.xml。

```
<!-- c3p0-config.xml 放在 src/main/resources 或 classpath 根目录 -->
```

```
<c3p0-config>
  <default-config>
    <property name="driverClass">com.mysql.cj.jdbc.Driver</property>
    <property
name="jdbcUrl">jdbc:mysql://localhost:3306/javaweb?useSSL=false&amp;serverTimezone=UTC&amp;characterEncoding=utf
8</property>
    <property name="user">root</property>
    <property name="password">123456</property>
    <property name="initialPoolSize">5</property>
    <property name="minPoolSize">5</property>
    <property name="maxPoolSize">20</property>
  </default-config>
</c3p0-config>
```

9.2 DBUtils 三大核心

核心	作用	常用写法
DbUtils 类	关闭资源、加载驱动等静态工具方法	DbUtils.closeQuietly(conn)
QueryRunner	简化 SQL 执行	new QueryRunner(dataSource); runner.query/update(...)
ResultSetHandler	把 ResultSet 转成 Bean、List、Map、单值	BeanHandler、BeanListHandler、ScalarHandler
Handler	**返回值**	**适合场景**
BeanHandler<>(User.class)	一个 JavaBean	登录查询单个用户
BeanListHandler<>(Student.class)	List<Student>	学生列表、商品列表
MapHandler / MapListHandler	Map 或 List<Map>	不想写 Bean 时临时用
ColumnListHandler	List	查询某一列所有值
ScalarHandler	Object/Long 等单值	count(*)、max、sum

9.3 C3P0 + DBUtils 工具类

```
public class DBUtil {
    private static DataSource dataSource = new ComboPooledDataSource();
```

```
    public static DataSource getDataSource() {
        return dataSource;
    }

    public static QueryRunner getQueryRunner() {
        return new QueryRunner(dataSource);
    }
}
```

9.4 DAO 模板

```
public class UserDao {
    public User login(String username, String password) {
        QueryRunner runner = DBUtil.getQueryRunner();
        String sql = "SELECT id, username, password FROM user WHERE username=? AND password=?";
        try {
            return runner.query(sql, new BeanHandler<>(User.class), username, password);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
}

public class StudentDao {
    public List<Student> getAllStudents() {
        QueryRunner runner = DBUtil.getQueryRunner();
        String sql = "SELECT id, name, snumber, sex, age FROM student";
        try {
            return runner.query(sql, new BeanListHandler<>(Student.class));
        } catch (SQLException e) {
            e.printStackTrace();
            return Collections.emptyList();
        }
    }
```

```
}
```

10 综合题模板

10.1 登录 + 查询 + JSP 列表

7. 建表：user(id, username, password)，student(id, name, snumber, sex, age)。
8. model：User、Student，字段私有、无参构造、getter/setter。
9. util：DBUtil 提供 DataSource/QueryRunner。
10. dao：UserDao.login 返回 User，StudentDao.getAllStudents 返回 List<Student>。
11. controller：LoginServlet 处理登录；StudentListServlet 检查 session 并查询学生列表。
12. view：login.jsp 表单提交到 /login；result.jsp 用 c:forEach 显示 ${studentList}。

```
// LoginServlet 重点
String username = req.getParameter("username");
String password = req.getParameter("password");
User user = new UserDao().login(username, password);
if (user != null) {
    req.getSession().setAttribute("loginUser", user);
    resp.sendRedirect(req.getContextPath() + "/studentList");
} else {
    req.setAttribute("errorMsg", "用户名或密码错误");
    req.getRequestDispatcher("/login.jsp").forward(req, resp);
}

// StudentListServlet 重点
HttpSession session = req.getSession(false);
if (session == null || session.getAttribute("loginUser") == null) {
    resp.sendRedirect(req.getContextPath() + "/login.jsp");
    return;
}
List<Student> students = new StudentDao().getAllStudents();
req.setAttribute("studentList", students);
req.getRequestDispatcher("/result.jsp").forward(req, resp);
```

10.2 购物车 Session + Cookie 模板

- Book 实体重写 equals/hashCode，按 id 判断同一本书。

- Cart 用 Map<Book,Integer> 保存商品和数量，提供 addBook、getTotalCount、getTotalPrice。
- 加入购物车时：从请求参数拿 id，找到 Book，取 session 中 cart，没有则创建，添加后重定向。
- Cookie 可保存 cartIds 或自动登录信息，要 setMaxAge 和 setPath。

```
HttpSession session = request.getSession();
Cart cart = (Cart) session.getAttribute("cart");
if (cart == null) {
    cart = new Cart();
    session.setAttribute("cart", cart);
}
cart.addBook(targetBook);
response.sendRedirect(request.getContextPath() + "/bookList");
```

11 易错点最后检查

易错点	正确写法/提醒
中文乱码	请求：request.setCharacterEncoding("UTF-8")；响应：response.setContentType("text/html;charset=UTF-8")
JSP 错误页拿不到 exception	错误页必须 <%@ page isErrorPage="true" %>
EL 取不到属性	检查是否放入正确作用域；Bean 是否有 getter；字段名大小写是否匹配
JSTL 不生效	检查 taglib URI、依赖 jar/API+实现包、Tomcat 版本 javax/jakarta 是否匹配
Filter 拦截后页面空白	忘记 chain.doFilter 或提前 return
redirect 后 request 数据没了	重定向是新请求；需要 session 或 URL 参数
forward 后地址栏不变	这是正常现象；forward 共享 request 域
上传拿不到文件	form 缺 enctype 或 Servlet 缺 @MultipartConfig
中文文件名下载乱码	Content-Disposition 使用 UTF-8 编码，filename*=UTF-8''...
SQL 注入	不要拼接用户输入，使用 PreparedStatement 或 QueryRunner 参数
DBUtils Bean 封装失败	列名要能匹配 JavaBean 属性；必要时 SQL 用别名：select snumber as snumber
C3P0 配置不生效	c3p0-config.xml 必须在 classpath 根目录，configName 不存在会走默认配置

12 实验报告对应索引

实验	主题	可复用考点
实验二	Servlet 技术的使用	ServletContext 共享数据、@WebServlet/@WebInitParam、RequestDispatcher include、Refresh 跳转
实验三	会话技术购物车	JavaBean、Session 保存购物车、Cookie 保存历史/自动信息、forward/redirect
实验四	JSP 技术	errorPage/isErrorPage、exception 隐式对象、jsp:include 动态包含
实验五	EL 和 JSTL	c:choose 表单分支、c:forEach 表格展示 List、EL 读取 Bean 属性
实验六	Servlet 高级特性	Filter 自动登录、CookieUtil、登录/退出、Session 检查
实验七	文件上传下载	@MultipartConfig、Part、getRealPath、URLEncoder、Content-Disposition
实验八	高级数据库技术	MVC、LoginServlet、StudentListServlet、C3P0、DBUtils、BeanHandler/BeanListHandler

13 一页背写骨架

```
// 1. Servlet 控制器
@WebServlet("/xxx")
public class XxxServlet extends HttpServlet {
  protected void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException {
    req.setCharacterEncoding("UTF-8");
    resp.setContentType("text/html;charset=UTF-8");
    // 取参数 -> 调 DAO/业务 -> 放域对象 -> forward 或 redirect
  }
}

// 2. JSP 显示集合
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<c:forEach items="${list}" var="item">
  ${item.name}
</c:forEach>

// 3. Filter
@WebFilter("/*")
public class XxxFilter implements Filter {
  public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
      throws IOException, ServletException {
    // 前置处理
    chain.doFilter(req, resp);
    // 后置处理
  }
}

// 4. DBUtils
QueryRunner runner = new QueryRunner(new ComboPooledDataSource());
runner.update("insert into user(username,password) values(?,?)", username, password);
User u = runner.query("select * from user where id=?", new BeanHandler<>(User.class), id);
List<Student> list = runner.query("select * from student", new BeanListHandler<>(Student.class));
```

14 往年卷选择填空速查

本节按往年卷选择/填空题补充。老师本次划的重点仍以 JavaBean、JSP、Servlet、EL、JSTL、Filter、上传下载、JDBC、C3P0、DBUtils 为主；Spring 相关先按低优先级看。

14.1 一眼选答案表

题干关键词	应选/应填	理由
不属于客户端应用技术	Servlet	HTML、CSS、DHTML 属于客户端/前端；Servlet 在服务器端运行
JSP 文件第一次被请求，容器把它转换成	Servlet	JSP 本质会被翻译/编译为 Servlet
JSP 页面 page 指令某属性可用多次，其他通常一次	import	page 指令中 import 可多次声明，也可逗号分隔多个类
局部属于 Servlet 的对象	ServletConfig	ServletConfig 是当前 Servlet 的配置对象；ServletContext 是整个 Web 应用共享
@WebServlet 中 urlPatterns 等价属性	value	@WebServlet("/x") 等价于 @WebServlet(urlPatterns="/x")
JDBC API 只有查询数据才会用到	ResultSet	executeQuery 返回 ResultSet；增删改用 executeUpdate
EL 和 JSP 都内置的隐式对象	pageContext	EL 有 pageContext；JSP 九大隐式对象也有 pageContext
c:when 与 c:otherwise 要和哪个标签组成条件结构	c:choose	JSTL 多分支：choose -> when -> otherwise
JSP Model2 主要技术	JSP + Servlet + JavaBean	JSP 显示，Servlet 控制，JavaBean/DAO 处理模型
AOP 可手动控制目标方法执行	Around Advice	环绕通知可在方法前后增强，并决定是否调用 proceed()

14.2 HTTP 与 Web 基础选择填空

知识点	标准答案

HTTP 请求消息组成	请求行、请求头、空行、请求体
HTTP 响应消息组成	状态行、响应头、空行、响应体
GET 参数位置	URL 后 query string，长度有限，适合查询
POST 参数位置	请求体，适合表单提交/上传
302/Location 或 sendRedirect	重定向，浏览器发起第二次请求，地址栏变化
RequestDispatcher.forward	请求转发，服务器内部跳转，地址栏不变，共享 request
请求包含 include	把其他资源输出包含进当前响应
Context 初始化参数标签	<context-param><param-name>...</param-name><param-value>...</param-value></context-param>
读取 Context 初始化参数	getServletContext().getInitParameter("参数名")
设置中文响应	response.setContentType("text/html;charset=UTF-8")
5 秒后自动跳转	response.setHeader("Refresh", "5;URL=https://www.cwnu.edu.cn/")

14.3 Spring 往年卷低优先级扩展

关键词	速查答案
Spring 核心	IoC/DI 和 AOP。IoC 把对象创建和依赖维护交给容器；AOP 面向切面增强横切逻辑
BeanFactory / ApplicationContext	ApplicationContext 是常用 Spring 容器，启动时加载配置并管理 Bean
DI 注入方式	setter 注入、构造器注入、注解注入
AOP 连接点 JoinPoint	程序执行点，常指方法调用
切点 Pointcut	匹配哪些连接点
通知 Advice	在切点处执行的增强代码
Before Advice	目标方法执行前
After Returning Advice	目标方法正常返回后
After Finally Advice	无论是否异常，最终都会执行

Around Advice	环绕目标方法，可手动控制是否执行目标方法
Spring JDBC	JdbcTemplate + DataSource，简化 JDBC 操作

15 往年卷编码题补充

15.1 web.xml 程序级初始化参数

```
<!-- Web 应用级初始化参数，位于 <web-app> 根元素内 -->
<context-param>
  <param-name>unitName</param-name>
  <param-value>CWNU</param-value>
</context-param>
<context-param>
  <param-name>address</param-name>
  <param-value>NanChong</param-value>
</context-param>

// Servlet 中读取
String unitName = getServletContext().getInitParameter("unitName");
String address = getServletContext().getInitParameter("address");
```

15.2 HttpServletResponse 常见两题

```
// 1. 解决中文输出乱码
response.setContentType("text/html;charset=UTF-8");
response.getWriter().write("中文内容");

// 2. 页面定时跳转，5 秒后自动跳转
response.setHeader("Refresh", "5;URL=https://www.cwnu.edu.cn/");
```

15.3 JavaBean 嵌套与集合属性

```
public class Employee {
    private String name;
    private String workNumber;
    public Employee() {}
    public String getName() { return name; }
```

```
    public void setName(String name) { this.name = name; }
    public String getWorkNumber() { return workNumber; }
    public void setWorkNumber(String workNumber) { this.workNumber = workNumber; }
}

public class Corp {
    private String corpName;
    private List<Employee> staff;
    public Corp() {}
    public String getCorpName() { return corpName; }
    public void setCorpName(String corpName) { this.corpName = corpName; }
    public List<Employee> getStaff() { return staff; }
    public void setStaff(List<Employee> staff) { this.staff = staff; }
}
```

15.4 Spring JDBC dataSource 配置

```
<!-- 往年卷扩展：本次老师未明确划 Spring，低优先级 -->
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
  <property name="url" value="jdbc:mysql://localhost:3306/javaweb?useSSL=false&amp;serverTimezone=UTC"/>
  <property name="username" value="root"/>
  <property name="password" value="123456"/>
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

15.5 Model2 + 原生 JDBC 综合题骨架

```
// DAO：查询 bookstore 全部记录
public ArrayList<Book> findAllBooks() {
    ArrayList<Book> list = new ArrayList<>();
    String sql = "select id, name, author, press, amount from bookstore";
    try (Connection conn = JDBCUtils.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql);
```

```
            ResultSet rs = ps.executeQuery()) {
            while (rs.next()) {
                Book b = new Book();
                b.setId(rs.getInt("id"));
                b.setName(rs.getString("name"));
                b.setAuthor(rs.getString("author"));
                b.setPress(rs.getString("press"));
                b.setAmount(rs.getInt("amount"));
                list.add(b);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }

    // Servlet：读取数据并放入 request 域
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        ArrayList<Book> books = new BookStoreDao().findAllBooks();
        request.setAttribute("books", books);
        request.getRequestDispatcher("/show.jsp").forward(request, response);
    }

    // show.jsp：EL + JSTL 显示
    <%@ taglib prefix="c" uri="jakarta.tags.core" %>
    <table>
      <tr><th>ID</th><th>书名</th><th>作者</th><th>出版社</th><th>数量</th></tr>
      <c:forEach items="${books}" var="b">
        <tr>
          <td>${b.id}</td><td>${b.name}</td><td>${b.author}</td>
          <td>${b.press}</td><td>${b.amount}</td>
        </tr>
      </c:forEach>
    </table>
```