JavaWeb 1-12 章大题预测与答题模板

Table of Contents

JavaWeb 1-12 章大题预测与答题模板

依据教学 PPT 第 1-12 章、老师给出的期末重点、往年卷题型整理。
目标：不是押原题，而是把“编程题/综合题可能怎么考、该写哪些代码、评分点在哪里”整理成开卷可翻的模板。

先看结论：最可能考的大题

优先级	可能题型	涉及章节	核心能力
高	JDBC/DBUtils 查询数据，Servlet 保存到域，JSP 用 EL/JSTL 表格显示	4、6、7、10、11	DAO + Servlet + JSP 综合
高	登录/注册功能：表单 → Servlet → 数据库 → Session/request → JSP	4、5、6、7、8、10、11	Model2/MVC 综合
高	Filter 编码过滤、登录检查、自动登录	5、9	Filter + Cookie/Session
高	文件上传/下载核心代码	4、9	multipart 表单、FileUpload、响应头
中高	C3P0 + DBUtils 配置与 QueryRunner 使用	10、11	连接池与简化 JDBC
中高	Ajax 用户名校验/局部刷新	12	jQuery Ajax + JSON
中	JSP/EL/JSTL 页面显示、条件判断、循环输出	6、7	View 层写法
中	Servlet 请求处理、转发、重定向、定时跳转	3、4	request/response
中	JavaBean 编写、jsp:useBean、MVC 分层说明	8	Bean 规范与 Model2
低中	XML、web.xml、context-param、Tomcat 基础配置	2、4	配置题
低中	HTTP 请求/响应报文分析	3	协议组成与状态码

大题通用答题主线

遇到综合题，先判断它要你写哪几层：

JSP 页面/HTML 表单

→ Servlet 接收请求

→ Service/JavaBean 处理业务

→ DAO/JDBC/DBUtils 操作数据库

→ request/session/application 保存结果

→ forward/redirect 到 JSP

→ JSP 用 EL/JSTL 展示

答题时不一定每层都写全，但要把关键得分点写出来：

- 表单：action、method、表单项 name。
- Servlet：@WebServlet、doGet/doPost、编码、取参数、调用 DAO、保存域对象、跳转。
- DAO：连接数据库、PreparedStatement 或 QueryRunner、处理 ResultSet。
- JSP：EL 取域对象，JSTL 判断/循环。
- 登录类题：成功存 session，失败存 request。
- 查询展示类题：查询结果一般存 request，然后 forward 到 JSP。

一、最高频综合题模板

1. 数据库列表查询 + Servlet + JSP 表格显示

可能怎么问

- 已知数据表 bookstore(id, name, author, press, amount) 和实体类 Book，实现查询所有图书并在 JSP 表格中显示。
- 已知表 user 或 goods，写 DAO 查询全部记录，Servlet 保存到域，JSP 使用 EL/JSTL 输出。
- 要求使用 JSP Model2，不能在 JSP 里直接写 JDBC。

答题结构

1. 写实体类 Book，属性与表字段对应。
2. 写 DAO 的 findAllBooks()，返回 List<Book>。
3. 写 Servlet 的 doGet()，调用 DAO，把结果放入 request。
4. forward 到 show.jsp。
5. JSP 用 <c:forEach> 循环输出表格。

DAO：JDBC 版本

```
public List<Book> findAllBooks() {
    List<Book> list = new ArrayList<Book>();
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try {
        conn = JDBCUtils.getConnection();
        String sql = "select id, name, author, press, amount from bookstore";
        ps = conn.prepareStatement(sql);
        rs = ps.executeQuery();

        while (rs.next()) {
            Book book = new Book();
            book.setId(rs.getInt("id"));
            book.setName(rs.getString("name"));
            book.setAuthor(rs.getString("author"));
            book.setPress(rs.getString("press"));
            book.setAmount(rs.getInt("amount"));
            list.add(book);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtils.release(rs, ps, conn);
    }
    return list;
}
```

DAO：DBUtils 版本

```
public List<Book> findAllBooks() throws SQLException {
    QueryRunner runner = new QueryRunner(C3P0Utils.getDataSource());
```

```
    String sql = "select id, name, author, press, amount from bookstore";
    return runner.query(sql, new BeanListHandler<Book>(Book.class));
}
```

Servlet

```
@WebServlet("/BookListServlet")
public class BookListServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse respons
e)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        BookDao dao = new BookDao();
        List<Book> books = dao.findAllBooks();
        request.setAttribute("books", books);
        request.getRequestDispatcher("/show.jsp").forward(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse respons
e)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

JSP：EL + JSTL 显示

```
<%@ page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<table border="1">
  <tr>
    <th>编号</th>
    <th>书名</th>
```

```
    <th>作者</th>
    <th>出版社</th>
    <th>数量</th>
  </tr>
  <c:forEach items="${books}" var="book">
    <tr>
      <td>${book.id}</td>
      <td>${book.name}</td>
      <td>${book.author}</td>
      <td>${book.press}</td>
      <td>${book.amount}</td>
    </tr>
  </c:forEach>
</table>
```

得分点

- DAO 不直接输出页面，只返回集合。
- Servlet 用 request.setAttribute() 保存查询结果。
- 查询展示用 forward，因为要保留 request 数据。
- JSP 用 ${book.name}，要求 Book 有 getName()。
- JSTL 要写 taglib。

---

2. 登录功能综合题

可能怎么问

- 编写用户登录功能，用户名密码正确进入首页，错误返回登录页并显示错误信息。
- 登录成功后在 Session 中保存用户信息。
- 使用 JDBC/DBUtils 查询数据库。

登录表单

```
<form action="${pageContext.request.contextPath}/LoginServlet" method="post">
  用户名：<input type="text" name="username">
  密码：<input type="password" name="password">
  <input type="submit" value="登录">
```

```
  <span>${msg}</span>
</form>
```

DAO

```
public User findUser(String username, String password) throws SQLException {
    QueryRunner runner = new QueryRunner(C3P0Utils.getDataSource());
    String sql = "select * from user where username = ? and password = ?";
    return runner.query(sql, new BeanHandler<User>(User.class), username, passwor
d);
}
```

Servlet

```
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse respons
e)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        String username = request.getParameter("username");
        String password = request.getParameter("password");

        UserDao dao = new UserDao();
        User user = dao.findUser(username, password);

        if (user != null) {
            request.getSession().setAttribute("user", user);
            response.sendRedirect(request.getContextPath() + "/index.jsp");
        } else {
            request.setAttribute("msg", "用户名或密码错误");
            request.getRequestDispatcher("/login.jsp").forward(request, response);
        }
```

```
    }
}
```

得分点

- 获取参数：request.getParameter()。
- 成功：用户对象放入 session。
- 失败：错误信息放入 request，转发回登录页。
- SQL 用 PreparedStatement 或 DBUtils 的占位符，不能拼接密码。

---

3. 注册功能 / MVC / JavaBean 综合题

可能怎么问

- 按 JSP Model2 思想实现用户注册。
- 创建表单 Bean，校验用户名、密码、邮箱。
- 注册失败返回注册页，成功进入成功页。

必写思路

register.jsp 提交表单

→ RegisterServlet 获取参数

→ RegisterFormBean 封装并校验表单

→ UserBean 封装用户信息

→ UserDao 保存到数据库

→ 成功存 session 或 request，失败回显错误

JavaBean 规范

```
public class User {
    private String username;
    private String password;
    private String email;

    public User() {
    }

    public String getUsername() {
        return username;
```

```
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

表单校验 Bean 思路

```
public boolean validate() {
    boolean flag = true;
    if (username == null || username.trim().equals("")) {
        errors.put("username", "用户名不能为空");
        flag = false;
    }
    if (email == null || !email.matches("[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\\.[a-zA-Z0-9_-]+)+")) {
        errors.put("email", "邮箱格式错误");
        flag = false;
    }
    return flag;
}
```

得分点

- JSP 是 View，不直接写数据库代码。
- Servlet 是 Controller，负责流程控制。
- JavaBean/DAO 是 Model，负责数据封装和业务/数据访问。
- 失败用 request.setAttribute("formBean", formBean) 回显错误。

二、章节可能大题

第 1 章 网页开发基础

可能考法

- 写一个 HTML 登录/注册/上传表单。

- 根据题目要求写 CSS 选择器或 JS 简单校验。
- 给出页面需求，写出表格、列表、超链接、图片、表单结构。

编程题模板：表单

```
<form action="LoginServlet" method="post">
  用户名：<input type="text" name="username">
  密码：<input type="password" name="password">
  性别：
  <input type="radio" name="sex" value="男">男
  <input type="radio" name="sex" value="女">女
  爱好：
  <input type="checkbox" name="hobby" value="java">Java
  <input type="submit" value="提交">
</form>
```

得分点

- 表单必须有 action 和 method。
- 后端要取值的表单项必须有 name。
- 单选/复选同组通常使用相同 name。

第 2 章 Java Web 概述

可能考法

- 写 XML 文档。
- 写 DTD 约束。
- 写 web.xml 中的全局初始化参数。
- 解释 Tomcat 目录或端口占用解决办法。

web.xml 全局初始化参数

题目例子：为 Web 应用创建程序级初始化参数 unitName=...、address=...。

```
<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 中获取：

```
String unitName = getServletContext().getInitParameter("unitName");
```

DTD 模板

```
<!ELEMENT 书架 (书+)>
<!ELEMENT 书 (书名,作者,售价)>
<!ELEMENT 书名 (#PCDATA)>
<!ELEMENT 作者 (#PCDATA)>
<!ELEMENT 售价 (#PCDATA)>
```

得分点

- XML 有且只有一个根元素。
- context-param 是整个 Web 应用共享，用 ServletContext 获取。
- Tomcat 默认端口是 8080，端口冲突改 server.xml 中 Connector 的 port。

第 3 章 HTTP 协议

可能考法

- 写出 HTTP 请求消息组成。
- 分析 GET/POST 区别。
- 根据场景写响应头，如编码、重定向、定时跳转、下载。
- 判断状态码含义。

HTTP 请求组成

请求行：请求方式 请求资源 协议版本
请求头：Host、User-Agent、Cookie、Content-Type...
空行
请求体：POST 提交的数据

HTTP 响应组成

状态行：协议版本 状态码 状态描述
响应头：Content-Type、Set-Cookie、Location...
空行
响应体：HTML/JSON/文件内容

response 常考代码

解决中文乱码：

```
response.setContentType("text/html;charset=UTF-8");
```

5 秒后跳转：

```
response.setHeader("refresh", "5;url=https://www.cwnu.edu.cn/");
```

重定向：

```
response.sendRedirect(request.getContextPath() + "/index.jsp");
```

得分点

- 302 是重定向，404 是资源不存在，500 是服务器异常。
- Content-Type 决定响应内容类型和编码。
- 定时跳转用 refresh 响应头。

第 4 章 Servlet 基础

可能考法

- 编写一个 Servlet，读取请求参数并响应。
- 写 @WebServlet 配置或 web.xml 配置。
- 使用 ServletConfig 读取当前 Servlet 初始化参数。
- 使用 ServletContext 读取全局参数或共享数据。
- 写转发/重定向区别或代码。

Servlet 基本模板

```
@WebServlet("/HelloServlet")
public class HelloServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse respons
```

```
e)
            throws ServletException, IOException {
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse respons
e)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        String username = request.getParameter("username");
        response.getWriter().write("欢迎你： " + username);
    }
}
```

转发和重定向

```
request.setAttribute("msg", "操作成功");
request.getRequestDispatcher("/success.jsp").forward(request, response);

response.sendRedirect(request.getContextPath() + "/success.jsp");
```

得分点

- 转发只有一次请求，地址栏不变，可保留 request 数据。
- 重定向两次请求，地址栏改变，request 数据丢失。
- @WebServlet(urlPatterns="/xxx") 和 @WebServlet("/xxx") 常见写法都可以。

第 5 章 会话及会话技术

可能考法

- Cookie 记录上次访问时间。
- Session 实现登录状态。
- 自动登录：Cookie 保存标识，Filter 或 Servlet 校验后写入 Session。
- 购物车：Session 保存购物车对象。

Cookie 模板

```
Cookie cookie = new Cookie("username", username);
cookie.setMaxAge(60 * 60 * 24 * 7);
cookie.setPath(request.getContextPath());
response.addCookie(cookie);
```

读取 Cookie：

```
Cookie[] cookies = request.getCookies();
for (Cookie cookie : cookies) {
    if ("username".equals(cookie.getName())) {
        String value = cookie.getValue();
    }
}
```

Session 登录模板

```
if (user != null) {
    request.getSession().setAttribute("user", user);
    response.sendRedirect(request.getContextPath() + "/index.jsp");
} else {
    request.setAttribute("msg", "登录失败");
    request.getRequestDispatcher("/login.jsp").forward(request, response);
}
```

退出登录

```
request.getSession().invalidate();
response.sendRedirect(request.getContextPath() + "/login.jsp");
```

得分点

- Cookie 存在客户端，Session 存在服务器端。
- 登录状态一般放 Session。
- setMaxAge(0) 可删除 Cookie。
- 自动登录不要明文保存密码，考试可写“实际项目中应加密”。

第 6 章 JSP 技术

可能考法

- 写 JSP 页面指令。
- 使用 JSP 隐式对象。
- 比较 include 指令和 <jsp:include>。
- 使用 <jsp:forward> 跳转。
- 根据 request/session 显示数据。

page 指令

```
<%@ page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8" %>
```

include 对比

项目	include 指令	<jsp:include> 动作
语法	<%@ include file="top.jsp" %>	<jsp:include page="top.jsp"/>
时机	翻译阶段静态包含	运行阶段动态包含
结果	合并成一个 Servlet	分别执行后合并结果
适用	静态页头页脚	动态内容

forward 动作

```
<jsp:forward page="login.jsp">
  <jsp:param name="msg" value="请先登录"/>
</jsp:forward>
```

得分点

- JSP 本质会被翻译成 Servlet。
- 九大隐式对象常考：request、response、session、application、out、pageContext、config、page、exception。
- 四大域：pageContext < request < session < application。

第 7 章 EL 表达式和 JSTL

可能考法

- 用 EL 获取 request/session 中的数据。
- 用 JSTL 判断是否登录。

- 用 <c:forEach> 输出集合。
- 用 <c:choose> 实现多分支。

EL 常用写法

```
${requestScope.msg}
${sessionScope.user.username}
${param.id}
${paramValues.hobby[0]}
${cookie.username.value}
${empty goodsList}
```

JSTL 判断

```
<c:choose>
  <c:when test="${sessionScope.user == null}">
    <a href="login.jsp">登录</a>
  </c:when>
  <c:otherwise>
    欢迎你，${sessionScope.user.username}
  </c:otherwise>
</c:choose>
```

JSTL 循环

```
<c:forEach items="${goodsList}" var="goods" varStatus="s">
  ${s.count} - ${goods.name} - ${goods.price}<br>
</c:forEach>
```

得分点

- JSTL 需要引入标签库：

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

- <c:when> 和 <c:otherwise> 不能单独使用，必须放在 <c:choose> 中。
- EL 默认按四大域从小到大查找，同名时建议写 requestScope 或 sessionScope。

第 8 章 JavaBean 与 JSP 开发模型

可能考法

- 编写一个 JavaBean 类。
- 用 jsp:useBean、jsp:setProperty、jsp:getProperty 操作 Bean。
- 说明 JSP Model1 和 Model2 区别。
- 按 MVC 思想分析一个注册/登录功能。

JavaBean 必背规范

1. 类通常是 public。
2. 必须有无参构造方法。
3. 属性私有化 private。
4. 提供 public 的 getter/setter。

JSP 操作 Bean

```
<jsp:useBean id="book" class="cn.itcast.Book" scope="request"/>
<jsp:setProperty name="book" property="*"/>

书名：<jsp:getProperty name="book" property="name"/>
```

Model2/MVC 角色

MVC	Java Web 中常见对应
Model	JavaBean、Service、DAO
View	JSP
Controller	Servlet

得分点

- property="*" 要求表单 name 与 JavaBean 属性名一致。
- Model1：JSP + JavaBean，JSP 仍承担控制流程。
- Model2：JSP + Servlet + JavaBean，Servlet 控制流程，更适合复杂项目。

第 9 章 Servlet 高级

可能考法

- 写编码过滤器。
- 写登录权限过滤器。
- 解释 Filter 生命周期和过滤器链。

- 写 Listener 监听 Web 应用启动/Session 创建。
- 写文件上传和下载关键代码。

编码 Filter

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

登录检查 Filter

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

        Object user = request.getSession().getAttribute("user");
        if (user == null) {
            response.sendRedirect(request.getContextPath() + "/login.jsp");
            return;
        }
        chain.doFilter(request, response);
    }
}
```

文件上传表单

```
<form action="UploadServlet" method="post" enctype="multipart/form-data">
  上传者：<input type="text" name="name">
  文件：<input type="file" name="myfile">
  <input type="submit" value="上传">
</form>
```

Commons-FileUpload 核心

```
DiskFileItemFactory factory = new DiskFileItemFactory();
ServletFileUpload upload = new ServletFileUpload(factory);
List<FileItem> items = upload.parseRequest(request);

for (FileItem item : items) {
    if (item.isFormField()) {
        String value = item.getString("UTF-8");
    } else {
        String filename = item.getName();
        item.write(new File(uploadPath, filename));
    }
}
```

文件下载响应头

```
response.setHeader("Content-Disposition", "attachment; filename=" + filename);
response.setContentType("application/x-msdownload");
```

得分点

- chain.doFilter() 表示放行，不写就不会到目标资源。
- Filter 生命周期：init()、doFilter()、destroy()。
- 上传表单必须是 post + multipart/form-data。
- 下载要设置 Content-Disposition: attachment。

第 10 章 JDBC

可能考法

- 写 JDBC 操作数据库的完整步骤。
- 用 PreparedStatement 实现查询/添加/删除/修改。
- 遍历 ResultSet 封装成 JavaBean。
- 对比 Statement 和 PreparedStatement。

JDBC 步骤

1. 加载数据库驱动
2. 获取 Connection
3. 编写 SQL
4. 创建 Statement/PreparedStatement
5. 执行 SQL
6. 处理 ResultSet
7. 释放资源

PreparedStatement 查询模板

```
String sql = "select * from user where username = ? and password = ?";
PreparedStatement ps = conn.prepareStatement(sql);
ps.setString(1, username);
ps.setString(2, password);
ResultSet rs = ps.executeQuery();

if (rs.next()) {
    User user = new User();
    user.setId(rs.getInt("id"));
    user.setUsername(rs.getString("username"));
}
```

增删改模板

```
String sql = "insert into user(username, password) values(?, ?)";
PreparedStatement ps = conn.prepareStatement(sql);
ps.setString(1, username);
```

```
ps.setString(2, password);
int rows = ps.executeUpdate();
```

得分点

- 查询用 executeQuery()。
- 增删改用 executeUpdate()。
- ResultSet 必须先 next()。
- PreparedStatement 下标从 1 开始。
- PreparedStatement 能防 SQL 注入。

第 11 章 数据库连接池与 DBUtils 工具

可能考法

- 写 C3P0 配置。
- 用 DataSource 获取连接。
- 用 QueryRunner 完成增删改查。
- 选择合适的 ResultSetHandler。

C3P0 配置模板

```
<c3p0-config>
  <default-config>
    <property name="driverClass">com.mysql.jdbc.Driver</property>
    <property name="jdbcUrl">jdbc:mysql://localhost:3306/test</property>
    <property name="user">root</property>
    <property name="password">123456</property>
  </default-config>
</c3p0-config>
```

工具类模板

```
public class C3P0Utils {
    private static DataSource dataSource = new ComboPooledDataSource();

    public static DataSource getDataSource() {
```

```
        return dataSource;
    }

    public static Connection getConnection() throws SQLException {
        return dataSource.getConnection();
    }
}
```

QueryRunner 查询

```
QueryRunner runner = new QueryRunner(C3P0Utils.getDataSource());
String sql = "select * from user where id = ?";
User user = runner.query(sql, new BeanHandler<User>(User.class), id);
```

QueryRunner 增删改

```
QueryRunner runner = new QueryRunner(C3P0Utils.getDataSource());
String sql = "delete from user where id = ?";
int rows = runner.update(sql, id);
```

Handler 选择

结果需求	使用
一行记录封装成 Bean	BeanHandler<T>
多行记录封装成 List	BeanListHandler<T>
查询某一列	ColumnListHandler
查询单个值，如 count	ScalarHandler

得分点

- C3P0 是连接池，DBUtils 是 JDBC 工具类。
- QueryRunner 配合 DataSource 可以自动获取和释放连接。
- 表字段名要能对应 JavaBean 属性名。
- 连接池中的 close() 通常表示归还连接。

第 12 章 Ajax

可能考法

- 使用 jQuery 发送 GET/POST/Ajax 请求。
- Ajax 检查用户名是否存在。
- Servlet 返回 JSON 或文本。
- 页面局部显示服务器响应结果。

Ajax 用户名校验模板

```
用户名：<input type="text" id="username" name="username">
<span id="msg"></span>

<script>
$("#username").blur(function () {
  $.ajax({
    url: "CheckUsernameServlet",
    type: "post",
    data: { username: $("#username").val() },
    dataType: "json",
    success: function (data) {
      if (data.exists) {
        $("#msg").text("用户名已存在");
      } else {
        $("#msg").text("用户名可以使用");
      }
    }
  });
});
</script>
```

Servlet 返回 JSON

```
@WebServlet("/CheckUsernameServlet")
public class CheckUsernameServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse respons
e)
```

```
        throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    response.setContentType("application/json;charset=UTF-8");

    String username = request.getParameter("username");
    boolean exists = "admin".equals(username);
    response.getWriter().write("{\"exists\":" + exists + "}");
  }
}
```

简单 GET/POST

```
$.get("UserServlet", { id: 1 }, function (data) {
  $("#result").html(data);
});

$.post("LoginServlet", { username: "tom", password: "123" }, function (data) {
  $("#msg").html(data);
});
```

得分点

- Ajax 是异步请求，不刷新整个页面。
- data 是发送到服务器的参数。
- success 是请求成功后的回调函数。
- 返回 JSON 时设置 dataType: "json"，服务器设置 application/json;charset=UTF-8。

三、开卷考试答题速查

看到题目关键词，马上定位

题目关键词	应写技术
表单提交	HTML form + Servlet getParameter()
中文乱码	request.setCharacterEncoding("UTF-8")、response.setContentType(...)

题目关键词	应写技术
页面跳转且保留数据	request.setAttribute() + forward
登录成功保存状态	session.setAttribute("user", user)
退出登录	session.invalidate()
自动登录	Cookie + Filter/Servlet 校验
统一编码	Filter
权限检查	Filter
查询列表显示	DAO 查询 List + request 域 + JSTL forEach
一行数据封装对象	BeanHandler
多行数据封装集合	BeanListHandler
查询总数	ScalarHandler
文件上传	multipart/form-data + ServletFileUpload
文件下载	Content-Disposition: attachment
局部刷新	Ajax

综合题最稳写法

如果题目要求“主要代码”，不要纠结完整项目，把下面几个关键块写清楚：

1. 实体类：属性 + getter/setter
2. DAO：SQL + PreparedStatement/QueryRunner + 返回对象/集合
3. Servlet：取参数 + 调 DAO + 存域 + 跳转
4. JSP：EL/JSTL 显示

容易扣分点

- 忘记 name，后端取不到表单参数。
- 查询结果放入 request 后使用了重定向，导致数据丢失。
- JSP 使用 JSTL 但忘记写 taglib。
- JavaBean 没有无参构造或 getter/setter。
- ResultSet 没写 while(rs.next())。
- PreparedStatement 参数下标从 1 开始，不是 0。
- 文件上传表单忘记 enctype="multipart/form-data"。

- Filter 中忘记 chain.doFilter()。
- Cookie 删除时只写 setMaxAge(0)，但路径和原 Cookie 不一致。

最后背这一条主线

浏览器请求 → Filter 可拦截 → Servlet 控制流程 → JavaBean/DAO 处理数据 → request/session 保存结果 → JSP + EL/JSTL 展示 → response 返回浏览器

大题本质就是把这条主线中的几段抽出来让你写。