JavaWeb 全课程零基础学习讲义

> 根据《Java Web程序设计任务教程（第2版）》PPT（第1章到第15章）整理。
>
> 说明：PPT 中的流程图、结构图和代码截图已经按含义转写成文字、流程和代码模板，方便直接阅读复习。

怎么用这份讲义

1. 先看“学习路线”，知道整门课从哪里开始、最后要做成什么。
2. 每章按顺序看：先读“一句话懂本章”，再看“知识点清单”，最后看“代码/流程模板”和“易错点”。
3. 如果是做题或开卷翻阅，优先查每章的“易错点”和代码模板。
4. 如果是从零开始学，建议一章一章敲小例子，不要只背名词。

目录

第1章 网页开发基础

一句话懂本章

网页 = HTML 负责结构，CSS 负责样式，JavaScript 负责行为，Bootstrap 帮你快速做出常见页面组件。

知识点清单

- HTML 基本结构：`<!DOCTYPE html>`、`html`、`head`、`body`。
- HTML 标签分类：双标签、单标签、注释标签。
- 常用 HTML 标签：段落、换行、文本样式、表格、表单、列表、超链接、图像。
- 表单重点：`form` 的 `action`、`method`；表单项的 `name`；`input` 的不同 `type`。
- CSS 作用：控制网页显示效果，如颜色、字体、边距、布局。
- CSS 引用方式：行内式、内嵌式、链入式、导入式。
- CSS 选择器：标签选择器、类选择器、ID选择器、后代选择器等。
- JavaScript 作用：让页面可以响应点击、输入、校验、动态修改内容。
- DOM：把 HTML 页面当成对象树，JS 可以查找和修改节点。
- BOM：浏览器对象模型，例如 `window`、`location`、`history`。
- Bootstrap：提供按钮、表单、导航、栅格布局等现成组件。

讲解

HTML 不负责“好不好看”，只负责告诉浏览器页面里有哪些东西。例如标题、段落、图片、表单都属于结构。CSS 是给这些结构“化妆”和排版的；JavaScript 是让页面“动起来”的，比如点击按钮后校验用户名是否为空。

表单是后面 Java Web 最常用的前端入口。浏览器提交表单时，后端通过表单项的 `name` 获取数据，所以 `name` 不能随便漏掉。`method="get"` 会把参数放到地址栏，适合查询；`method="post"` 会把参数放到请求体，适合登录、注册、上传等。

代码模板

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>登录页面</title>
  <link rel="stylesheet" href="css/bootstrap.css">
</head>
<body>
  <form action="LoginServlet" method="post">
    用户名: <input type="text" name="username">
    密码: <input type="password" name="password">
    <input type="submit" value="登录">
  </form>
</body>
</html>
```

易错点

- 表单项没有 `name`，后端 `request.getParameter()` 就取不到值。
- `id` 一般页面唯一，`class` 可以重复使用。

- CSS 四种引用方式中，项目里最常用的是链入式：`<link rel="stylesheet" href="...">`。
- JavaScript 获取的是浏览器里的 DOM 对象，不是服务器里的 Java 对象。

第2章 Java Web概述

一句话懂本章

Java Web 就是用 Java 写能部署在服务器上的动态网站；Tomcat 是运行 Servlet/JSP 的服务器，XML 常用于配置和数据描述。

知识点清单

- XML：可扩展标记语言，用于存储和传输数据，也常用于配置。
- XML 与 HTML 区别：HTML 负责显示，XML 负责描述数据；XML 严格区分大小写且只能有一个根元素。
- XML 基本语法：文档声明、元素、属性、注释。
- DTD 约束：规定 XML 中能出现哪些元素、元素顺序、属性等。
- Schema 约束：比 DTD 更强，使用 XML 语法，支持数据类型和命名空间。
- C/S 架构：客户端/服务器，需要安装客户端程序。
- B/S 架构：浏览器/服务器，用户通过浏览器访问。
- Tomcat：常用 Java Web 服务器和 Servlet 容器。
- Tomcat 目录：`bin`、`conf`、`lib`、`logs`、`webapps`、`work`。
- 常见问题：`JAVA_HOME` 配置错误、8080 端口被占用。
- IDEA 配置 Tomcat：创建 Web 项目、配置输出目录、配置依赖 Jar、部署 Artifact、设置 Application context。

讲解

XML 不是用来显示页面的，而是用来描述数据结构。比如“书架里有多本书，每本书有书名、作者、售价”，这种层级结构用 XML 很自然。DTD 和 Schema 的作用是给 XML 加规则，防止 XML 写得乱七八糟。

Tomcat 可以理解成 Java Web 程序的运行环境。浏览器发请求给 Tomcat，Tomcat 再根据路径找到 Servlet/JSP 执行，然后把响应返回给浏览器。后面学 Servlet、JSP 都离不开它。

XML 例子

```xml
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <name>Java Web程序设计</name>
  <price unit="元">68</price>
</book>
```

DTD 例子

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

Tomcat 目录速记

目录	作用
`bin`	启动/关闭脚本
`conf`	配置文件，如 `server.xml`、`web.xml`
`lib`	Tomcat 和 Web 应用可能用到的 Jar
`logs`	日志
`webapps`	Web 项目发布目录
`work`	JSP 编译后的 Servlet 文件和 class 文件

易错点

- XML 必须有且只能有一个根元素。
- Tomcat 启动后访问地址通常是 `http://localhost:8080`。
- 8080 被占用时，可在 `conf/server.xml` 中修改 `Connector` 的 `port`。
- `webapps` 是部署项目的位置，`work` 是 JSP 编译后的工作目录。

第3章 HTTP协议

一句话懂本章

HTTP 是浏览器和服务器说话的规则：浏览器发请求，服务器给响应。

知识点清单

- HTTP 特点：基于请求/响应模型，默认无状态。
- HTTP 请求消息：请求行、请求头、空行、请求体。
- 请求行：请求方式、请求资源路径、协议版本。
- 常用请求方式：`GET`、`POST`。
- 常见请求头：`Host`、`User-Agent`、`Accept`、`Referer`、`Cookie`、`Content-Type`。
- HTTP 响应消息：状态行、响应头、空行、响应体。
- 状态行：协议版本、状态码、状态描述。
- 常见状态码：`200`、`302`、`304`、`404`、`500`。
- HTTP/1.0 与 HTTP/1.1：HTTP/1.1 默认支持持久连接。

讲解

HTTP 无状态的意思是：服务器不会天然记得“上一次请求是谁”。所以后面才需要 Cookie 和 Session 来记住用户登录状态。

`GET` 常用于查询，参数会显示在 URL 后面；`POST` 常用于提交表单，参数放在请求体中。两者不是“谁更高级”，而是适用场景不同。

请求消息长这样

```
POST /chapter04/LoginServlet HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded

username=tom&password=123456
```

响应消息长这样

```
HTTP/1.1 200 OK
Content-Type: text/html;charset=UTF-8

<html>...</html>
```

状态码速记

状态码	含义
`200`	请求成功
`302`	重定向
`304`	使用缓存
`404`	资源不存在
`500`	服务器内部错误

易错点

- `404` 多半是路径错或资源没部署；`500` 多半是服务器代码异常。
- `GET` 参数在地址栏，长度有限且不适合传密码。
- HTTP 协议无状态，不代表服务器不能保存状态，而是要借助 Cookie/Session 等技术保存。

第4章 Servlet基础

一句话懂本章

Servlet 是运行在服务器端的 Java 程序，专门接收请求、处理业务、生成响应。

知识点清单

- Servlet 概念：动态 Web 资源，运行在 Servlet 容器中。
- Servlet 接口方法：`init()`、`service()`、`destroy()`。
- 生命周期：创建/初始化、处理请求、销毁。
- 开发方式：实现 `Servlet`，继承 `GenericServlet`，常用继承 `HttpServlet`。
- 配置方式：`web.xml` 或 `@WebServlet` 注解。
- `ServletConfig`：获取单个 Servlet 的初始化参数。
- `ServletContext`：代表整个 Web 应用，可获取全局参数、真实路径、共享数据。
- `HttpServletRequest`：获取请求参数、请求头、请求路径、请求域数据。
- `HttpServletResponse`：设置响应内容、编码、重定向、输出数据。
- 请求转发：`request.getRequestDispatcher(...).forward(request, response)`。
- 重定向：`response.sendRedirect(...)`。

讲解

浏览器不能直接运行 Java 类，Tomcat 可以。浏览器请求某个 URL，Tomcat 根据 URL 找到对应的 Servlet，调用它的方法。`HttpServlet` 已经帮我们区分了 HTTP 请求方式，所以一般重写 `doGet()` 和 `doPost()`。

`request` 是“这一次请求带来的东西”，例如参数、请求头、请求域；`response` 是“服务器要返回的东西”，例如 HTML、重定向、下载文件。

Servlet 模板

```
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        doPost(request, response);
    }

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

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

转发与重定向对比

项目	请求转发 `forward`	重定向 `sendRedirect`
发起者	服务器内部跳转	浏览器重新发请求
请求次数	1 次	2 次
地址栏	不变	改变
request 数据	可以保留	不能保留
可跳转范围	当前 Web 应用内部	可以跳到外部网站

易错点

- `init()` 不是每次请求都执行，通常 Servlet 创建时执行一次。
- `ServletContext` 是整个 Web 应用共享，`ServletConfig` 是单个 Servlet 的配置。
- 转发适合把数据带到 JSP 展示；重定向适合登录成功后跳到新页面，防止表单重复提交。
- 中文乱码通常要同时注意请求编码和响应编码。

第5章 会话及会话技术

一句话懂本章

Cookie 和 Session 用来解决 HTTP 无状态问题，让服务器“记住”用户。

知识点清单

- 会话：用户打开浏览器访问网站，到结束访问之间的一系列请求和响应。
- Cookie：服务器发给浏览器保存的小数据，下次请求时浏览器自动带回。
- Cookie API：构造方法、`setMaxAge()`、`setPath()`、`getName()`、`getValue()`。
- Session：服务器端保存用户状态的数据空间，每个浏览器会话通常对应一个 Session。
- 获取 Session：`request.getSession()`。
- Session API：`setAttribute()`、`getAttribute()`、`removeAttribute()`、`invalidate()`。
- Session 生命周期：创建、生效、超时失效、手动销毁。
- 常见应用：登录状态、购物车、验证码、上次访问时间。

讲解

Cookie 放在浏览器端，Session 放在服务器端。浏览器通常会保存一个名为 `JSESSIONID` 的 Cookie，服务器用它找到对应 Session。也就是说，Session 的识别常常依赖 Cookie。

登录功能常见做法：登录成功后，把用户对象放到 Session；访问需要登录的页面时，先判断 Session 中有没有用户。

Cookie 模板

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

Session 模板

```
// 登录成功
request.getSession().setAttribute("user", user);

// 判断是否登录
User user = (User) request.getSession().getAttribute("user");
if (user == null) {
    response.sendRedirect(request.getContextPath() + "/login.jsp");
}

// 退出登录
request.getSession().invalidate();
```

Cookie 与 Session 对比

项目	Cookie	Session
保存位置	浏览器端	服务器端
安全性	较低，不适合直接存密码	较高
数据大小	较小	相对更灵活
生命周期	可通过 `setMaxAge` 设置	默认会话期间，可设置超时
常见用途	自动登录、上次访问时间	登录状态、购物车

易错点

- Cookie 存密码要加密，不能明文保存。
- Session 不是浏览器关闭一定立刻销毁，通常是服务器端超时或调用 `invalidate()` 后销毁。
- `request` 域只在一次请求内有效；`session` 域跨多次请求有效。

第6章 JSP技术

一句话懂本章

JSP 本质上会被 Tomcat 翻译成 Servlet，用来更方便地写动态页面。

知识点清单

- JSP 作用：在 HTML 页面中嵌入动态内容。
- JSP 运行原理：JSP 文件先翻译成 Servlet，再编译成 class，最后执行。
- JSP 基本语法：模板元素、脚本片段、声明、表达式。
- JSP 注释：HTML 注释、JSP 隐藏注释、动态注释。
- JSP 指令：`page`、`include`、`taglib`。
- JSP 动作元素：`jsp:include`、`jsp:forward`、`jsp:param`。
- JSP 九大隐式对象：`request`、`response`、`session`、`application`、`out`、`pageContext`、`config`、`page`、`exception`。
- 四大域对象：`pageContext`、`request`、`session`、`application`。

讲解

JSP 适合写显示页面，不适合堆大量 Java 业务代码。早期写法会在 JSP 里写 `<% Java代码 %>`，但后面更推荐用 EL/JSTL 展示数据，把控制逻辑放到 Servlet，把数据封装到 JavaBean。

隐式对象就是 JSP 页面里不用自己创建、可以直接使用的对象。比如 `request` 获取请求参数，`session` 保存登录状态，`application` 对应整个 Web 应用。

JSP 基本写法

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

<%-- JSP隐藏注释：客户端看不到 --%>

<%
    String username = request.getParameter("username");
%>

欢迎你：<%= username %>
```

JSP 动作元素

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

四大域对象速记

域对象	有效范围
`pageContext`	当前 JSP 页面
`request`	一次请求
`session`	一次会话
`application`	整个 Web 应用

易错点

- JSP 第一次访问慢，是因为要翻译和编译。
- `<%@ include %>` 是静态包含，先合并再编译；`<jsp:include>` 是动态包含，运行时包含。
- `exception` 隐式对象通常只在错误页面中可用。
- 开发中尽量减少 JSP 脚本片段，把显示逻辑交给 EL/JSTL。

第7章 EL表达式和JSTL

一句话懂本章

EL 用来更简单地取数据，JSTL 用标签完成判断、循环等页面逻辑，目的都是减少 JSP 里的 Java 代码。

知识点清单

- EL 基本语法：`${表达式}`。
- EL 作用：获取域对象数据、请求参数、Cookie、初始化参数等。
- EL 标识符：变量名要符合命名规则，不能使用关键字。
- EL 访问数据：点运算符 `.` 和方括号 `[]`。
- EL 运算符：算术、关系、逻辑、`empty`。
- EL 隐式对象：`pageScope`、`requestScope`、`sessionScope`、`applicationScope`、`param`、`paramValues`、`cookie`、`initParam`、`pageContext`。
- JSTL：JSP 标准标签库，需要导入 Jar 并用 `taglib` 声明。
- Core 核心标签：`c:out`、`c:set`、`c:remove`、`c:if`、`c:choose`、`c:forEach`、`c:url`、`c:redirect`。

讲解

EL 取值时，如果不写作用域，会按从小到大的范围查找：`pageScope → requestScope → sessionScope → applicationScope`。如果多个域里都有同名数据，建议写清楚作用域，例如 `${sessionScope.user.username}`。

JSTL 的核心标签库常用于页面判断和循环。例如登录后显示用户名，未登录显示登录链接；商品列表循环输出到页面。这些都不需要再写 `<% for (...) { %>`。

EL 常见写法

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

JSTL 使用模板

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

<c:if test="${sessionScope.user ≠ null}">
    欢迎你，${sessionScope.user.username}
</c:if>

<c:choose>
    <c:when test="${empty goodsList}">暂无商品</c:when>
    <c:otherwise>
        <c:forEach items="${goodsList}" var="goods">
            ${goods.name} - ${goods.price}<br>
        </c:forEach>
    </c:otherwise>
</c:choose>
```

易错点

- `${user.name}` 本质上会调用 `getName()`，所以 JavaBean 要有规范 getter。
- `.` 适合普通属性名；属性名包含特殊字符时用 `[]`。
- `param` 取单个参数，`paramValues` 取多个同名参数。
- `empty` 可以判断 `null`、空字符串、空集合。

第8章 JavaBean与JSP开发模型

一句话懂本章

JavaBean 用来封装数据和业务逻辑；JSP Model2/MVC 用来把页面、控制、数据处理分开。

知识点清单

- JavaBean 概念：遵循特定写法的 Java 类，可被 JSP/Servlet 调用。
- JavaBean 规范：无参构造、属性私有、提供 public getter/setter。
- JavaBean 类型：可视化 JavaBean、非可视化 JavaBean；Java Web 主要使用非可视化。
- JSP 操作 JavaBean：`jsp:useBean`、`jsp:setProperty`、`jsp:getProperty`。
- `property="*"`：表单项 name 与 Bean 属性名一致时，可自动批量赋值。
- JSP 早期模型：JSP 同时负责页面显示、流程控制、业务逻辑，耦合高。
- JSP Model1：JSP + JavaBean，业务逻辑部分抽到 Bean，但 JSP 仍负责流程控制。
- JSP Model2：JSP + Servlet + JavaBean，接近 MVC。
- MVC：Model 模型、View 视图、Controller 控制器。

讲解

JavaBean 最重要的价值是“封装”。比如用户信息不要散落在多个字符串变量里，而是封装成 `User` 对象。后面 EL 访问 `${user.username}`，也是依赖 JavaBean 的 getter 方法。

Model2/MVC 是 Java Web 项目真正走向分层的关键：Servlet 做控制器，负责接收请求、调用业务、决定转发到哪个 JSP；JSP 做视图，只负责显示；JavaBean/DAO/Service 做模型，负责数据和业务。

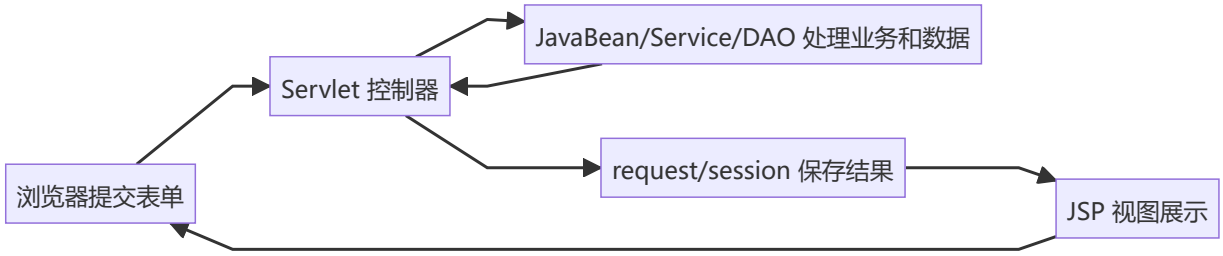
JavaBean 模板

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

    public User() {
    }

    public String getUsername() {
        return username;
    }

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

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

JSP 操作 JavaBean

```jsp
<jsp:useBean id="user" class="cn.itcast.User" scope="request"/>
<jsp:setProperty name="user" property="username" value="tom"/>
用户名: <jsp:getProperty name="user" property="username"/>
```

MVC 流程



易错点

- JavaBean 属性名和 getter/setter 命名要对应：`username` → `getUsername()/setUsername()`。
- `jsp:setProperty property="*"` 要求表单 `name` 与 Bean 属性名一致。
- Model1 不是完全错误，只是不适合复杂项目；Model2 更适合维护。
- MVC 中 Servlet 不是用来写页面 HTML 的，JSP 也不应该负责数据库操作。

第9章 Servlet高级

一句话懂本章

Filter 负责拦截请求/响应，Listener 负责监听事件，文件上传下载和 Servlet 3.0 注解让 Web 功能更完整。

知识点清单

- Filter 作用：在请求到达目标资源前、响应返回客户端前进行拦截处理。
- Filter API：`Filter`、`FilterConfig`、`FilterChain`。
- Filter 生命周期：`init()`、`doFilter()`、`destroy()`。
- `chain.doFilter(request, response)`：放行到下一个过滤器或目标资源。
- Filter 映射：`urlPatterns`、通配符、`dispatcherTypes`。
- Filter 链：多个 Filter 拦截同一 URL，按顺序执行。
- Listener 作用：监听 `ServletContext`、`HttpSession`、`ServletRequest` 的创建、销毁和属性变化。
- 常用 Listener 接口：`ServletContextListener`、`HttpSessionListener`、`ServletRequestListener` 等。
- Servlet 3.0 新特性：`@WebServlet`、`@WebFilter`、`@WebListener`、`@MultipartConfig`、异步处理。
- 文件上传：表单必须 `method="post"` 且 `enctype="multipart/form-data"`。
- Commons-FileUpload：`FileItem`、`DiskFileItemFactory`、`ServletFileUpload`。

- 文件下载：设置 `Content-Disposition` 和 `Content-Type`，用 IO 流输出。

Filter 讲解

Filter 像“门卫”，所有符合映射规则的请求都会先经过它。常见用途：登录检查、统一编码、自动登录、权限控制、敏感词过滤。

如果 `doFilter()` 中不调用 `chain.doFilter()`，请求就不会继续到达目标 Servlet/JSP。这个点非常容易考。

Filter 模板

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

Filter 链执行顺序

```
请求进入 → Filter1 前置代码 → Filter2 前置代码 → Servlet
        ← Filter2 后置代码 ← Filter1 后置代码 ← 响应返回
```

Listener 讲解

Listener 适合在对象生命周期变化时自动执行代码。例如 Web 应用启动时初始化资源，Session 创建/销毁时统计在线人数，请求创建/销毁时记录日志。

```java
@WebListener
public class MyListener implements ServletContextListener {
    public void contextInitialized(ServletContextEvent event) {
        System.out.println("Web应用启动");
    }

    public void contextDestroyed(ServletContextEvent event) {
        System.out.println("Web应用关闭");
    }
}
```

文件上传模板

```html
<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>
```

```java
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(uploadDir, filename));
    }
}
```

文件下载核心

```
response.setHeader("Content-Disposition", "attachment;
filename=1.png");
response.setContentType("application/x-msdownload");
```

易错点

- `Filter` 的 `init()` 创建时执行一次，`doFilter()` 每次匹配请求都会执行。
- `chain.doFilter()` 前是请求阶段，后是响应回来的阶段。
- `@WebFilter` 多个过滤器使用注解时，执行顺序不如 `web.xml` 那样容易手动控制，PPT 中强调可按类名自然顺序理解。
- 文件上传表单必须设置 `enctype="multipart/form-data"`，否则后端拿不到文件内容。
- 下载不是简单链接文件，而是通过响应头告诉浏览器“按附件下载”。

第10章 JDBC

一句话懂本章

JDBC 是 Java 连接数据库的标准 API，核心流程是加载驱动、建立连接、执行 SQL、处理结果、释放资源。

知识点清单

- JDBC 概念：Java Database Connectivity，Java 访问数据库的规范。
- 常用 API：`Driver`、`DriverManager`、`Connection`、`Statement`、`PreparedStatement`、`ResultSet`。
- JDBC 操作步骤：加载驱动、获取连接、编写 SQL、创建执行对象、执行 SQL、处理结果集、关闭资源。
- `Statement`：执行静态 SQL。
- `PreparedStatement`：预编译 SQL，使用 `?` 占位符，防 SQL 注入，效率更好。
- `ResultSet`：保存查询结果，使用 `next()` 移动游标。
- CRUD：增、删、改通常用 `executeUpdate()`；查用 `executeQuery()`。

讲解

JDBC 是 Java 和数据库之间的“通用插头”。Java 不直接关心底层是 MySQL 还是其他数据库，而是通过 JDBC API 调用数据库驱动。

`PreparedStatement` 是重点。它先把 SQL 结构固定下来，再给参数赋值。用户输入只会被当作参数，不会被拼接成 SQL 代码，所以能防止 SQL 注入。

JDBC 查询模板

```
Connection conn = null;
PreparedStatement ps = null;
ResultSet rs = null;

try {
    Class.forName("com.mysql.jdbc.Driver");
    conn = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=utf8",
            "root",
            "password");

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

    rs = ps.executeQuery();
    if (rs.next()) {
        int id = rs.getInt("id");
        String name = rs.getString("username");
    }
} finally {
    if (rs ≠ null) rs.close();
    if (ps ≠ null) ps.close();
    if (conn ≠ null) conn.close();
}
```

Statement 与 PreparedStatement 对比

项目	Statement	PreparedStatement
SQL 写法	字符串拼接	`?` 占位符
安全性	容易 SQL 注入	可防 SQL 注入
性能	每次编译 SQL	可预编译
推荐程度	少用	常用

易错点

- `ResultSet` 初始游标在第一行之前，必须先 `rs.next()`。
- 增删改用 `executeUpdate()`，返回影响行数；查询用 `executeQuery()`，返回结果集。
- 数据库连接、执行对象、结果集都要关闭。
- SQL 中的第一个 `?` 对应 `setXxx(1, value)`，下标从 1 开始。

第11章 数据库连接池与DBUtils工具

一句话懂本章

连接池负责复用数据库连接，DBUtils 负责简化 JDBC 代码。

知识点清单

- 数据库连接池：提前创建一批连接，用完归还，避免频繁创建/销毁连接。
- `DataSource`：数据源接口，获取连接池中的连接。
- DBCP：一种数据库连接池实现。
- C3P0：常用开源数据库连接池，核心类 `ComboPooledDataSource`。
- C3P0 配置：可通过代码或 `c3p0-config.xml` 配置驱动、URL、用户名、密码、连接数。
- DBUtils：Apache 提供的 JDBC 工具库。
- `DbUtils` 类：关闭资源、提交/回滚等工具方法。
- `QueryRunner`：执行 SQL 的核心类，提供 `query()`、`update()`。
- `ResultSetHandler`：把 `ResultSet` 转换成需要的 Java 对象。
- 常用 Handler：`BeanHandler`、`BeanListHandler`、`ColumnListHandler`、`ScalarHandler`。

讲解

普通 JDBC 每次都创建数据库连接，成本很高。连接池的思想是“借连接、用连接、还连接”，程序调用 `close()` 时通常不是物理关闭，而是归还到池中。

DBUtils 帮你省掉大量重复代码，例如创建 `PreparedStatement`、遍历 `ResultSet`、关闭资源。它不是数据库，也不是连接池，而是 JDBC 工具类库。

C3P0 数据源模板

```
ComboPooledDataSource dataSource = new ComboPooledDataSource();
Connection conn = dataSource.getConnection();
```

常见 `c3p0-config.xml` 思路：

```
<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">password</property>
  </default-config>
</c3p0-config>
```

DBUtils 查询模板

```
QueryRunner runner = new QueryRunner(dataSource);

String sql = "select * from user where id = ?";
User user = runner.query(sql, new BeanHandler<User>(User.class), 1);

List<User> users = runner.query(
        "select * from user",
        new BeanListHandler<User>(User.class));
```

常用 Handler 速记

Handler	作用
`BeanHandler<T>`	查询一行，封装成一个 JavaBean
`BeanListHandler<T>`	查询多行，封装成 List
`ColumnListHandler`	查询某一列，封装成 List
`ScalarHandler`	查询单个值，如总数

易错点

- 连接池中的 `conn.close()` 通常表示归还连接，不是彻底关闭数据库连接。
- `BeanHandler` 要求表字段名和 JavaBean 属性名能对应。
- `QueryRunner.update()` 可执行增删改。
- C3P0 是连接池，DBUtils 是工具库，两者常配合使用，但不是同一个东西。

第12章 Ajax

一句话懂本章

Ajax 让页面不整体刷新，也能和服务器交换数据。

知识点清单

- Ajax：Asynchronous JavaScript and XML，异步请求技术。
- 作用：局部刷新，提高交互体验。
- jQuery 基础：选择器、事件、DOM 操作。
- jQuery 常用 Ajax 方法：`load()`、`$.get()`、`$.post()`、`$.ajax()`。
- GET/POST 区别：GET 常用于查询，POST 常用于提交。
- JSON：轻量级数据交换格式，常用于前后端传输对象和数组。
- Ajax 基础流程：前端发异步请求，服务器返回数据，前端局部更新页面。

讲解

没有 Ajax 时，用户点一下按钮，整个页面可能都要重新加载。有了 Ajax，页面可以只把需要的数据发给服务器，再把返回结果局部显示出来。例如注册时检查用户名是否已存在。

现在 Ajax 常用 JSON，而不是 XML。JSON 更贴近 JavaScript 对象，写法简洁。

jQuery Ajax 模板

```
$.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("可以使用");
    }
  },
  error: function () {
    $("#msg").text("请求失败");
  }
});
```

JSON 例子

```
{
  "id": 1,
  "username": "tom",
  "roles": ["user", "admin"]
}
```

易错点

- Ajax 是浏览器在后台发请求，不代表没有请求。
- `dataType` 表示希望把服务器返回结果按什么格式解析。
- GET 参数通常拼在 URL 上，POST 参数放在请求体中。

- Ajax 返回 JSON 时，服务器响应类型最好设置为 `application/json;charset=UTF-8`。

第13章 网上蛋糕商城-项目搭建

一句话懂本章

这一章把前面零散技术组织成一个完整项目：先分析需求，再设计数据库和项目结构。

知识点清单

- 项目需求：网上蛋糕商城，包含前台购物和后台管理。
- 前台功能：用户注册、登录、购物车、商品分类查询、商品搜索等。
- 后台功能：商品管理、订单管理、客户管理、商品类目管理等。
- 数据库设计：根据 E-R 图设计数据表。
- 项目主要表：`user`、`goods`、`order`、`orderitem`、`type`、`recommend`。
- 环境搭建：创建 Web 项目、导入 Jar、配置数据库、配置 Tomcat。
- 分层思想：实体类、DAO、Service、Servlet、JSP 页面。

讲解

项目不是一上来就写代码。先看需求：用户要买蛋糕，管理员要维护商品和订单。再看有哪些对象：用户、商品、订单、订单项、分类、推荐。最后把对象落到数据库表里。

订单通常要拆成 `order` 和 `orderitem`。`order` 记录一次订单的整体信息，如用户、总价、状态；`orderitem` 记录这个订单里每一种商品买了几个。

项目结构理解

```
cn.itcast
  entity        实体类：User、Goods、Order...
  dao           数据访问：写SQL
  service       业务逻辑：注册、登录、下单
  servlet       控制器：接收请求，调用service，转发页面
web
  index.jsp
  login.jsp
  register.jsp
  goods_list.jsp
WEB-INF
  lib           第三方Jar包
```

数据表速记

表	作用
`user`	用户/客户信息
`goods`	商品信息
`type`	商品分类
`order`	订单主表
`orderitem`	订单明细
`recommend`	商品推荐关系或推荐类型

易错点

- 需求分析决定功能边界，数据库设计决定后面代码好不好写。

- 一对多关系常通过外键表达，例如一个分类对应多个商品。
- 购物车不一定一开始就入库，常可先放 Session。
- 项目搭建时 Jar 包既要复制到 `WEB-INF/lib`，也要在 IDEA 中配置依赖。

第14章 网上蛋糕商城-前端开发

一句话懂本章

前台开发就是让普通用户能注册、登录、浏览商品、搜索商品、加入购物车并下单。

知识点清单

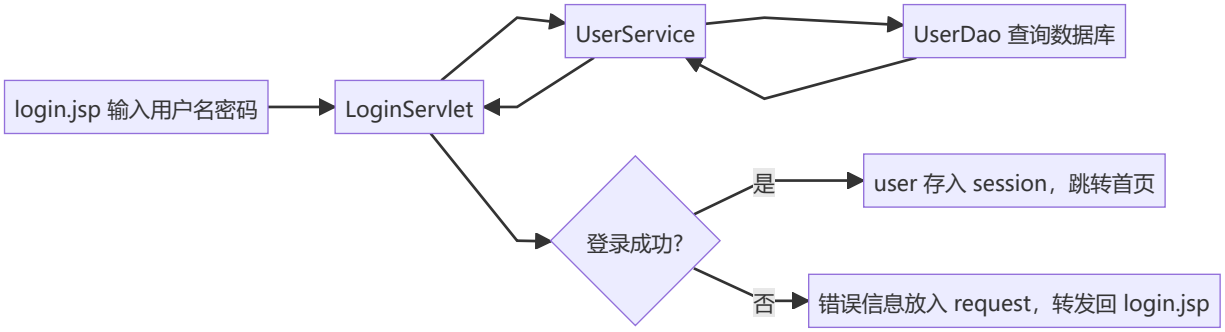
- 用户注册：表单提交、参数校验、写入数据库、失败回显错误。
- 用户登录：查询用户、保存登录状态到 Session。
- 购物车：添加商品、修改数量、删除商品、计算总价。
- 商品分类查询：根据分类 ID 查询商品列表。
- 商品搜索：根据关键字模糊查询商品。
- 前台流程：JSP 表单/链接 -> Servlet -> Service -> DAO -> 数据库 -> JSP 展示。

讲解

前台功能本质上都在围绕“用户请求”展开。比如用户点击“加入购物车”，浏览器提交商品 ID，Servlet 接收 ID，Service 处理购物车逻辑，最后把结果放到 Session 或 request，再转到 JSP 展示。

购物车常见设计：Session 中存一个 `Map<商品ID, 购物项>`。购物项里包含商品对象、数量、小计。这样同一个商品再次加入时，只需要数量 +1。

登录流程



购物车核心思路

```java
Cart cart = (Cart) request.getSession().getAttribute("cart");
if (cart == null) {
    cart = new Cart();
    request.getSession().setAttribute("cart", cart);
}
cart.addGoods(goods);
```

易错点

- 登录成功用 Session 保存用户；登录失败用 request 保存错误消息并转发回登录页。
- 搜索通常使用 SQL 的 `like`，参数要用 `PreparedStatement`，不要字符串拼接。
- 购物车放 Session 时，刷新页面不能丢；但用户退出或 Session 失效后会消失。
- 前台 JSP 不直接操作数据库，数据库操作放 DAO。

第15章 网上蛋糕商城-后台开发

一句话懂本章

后台开发是给管理员用的，用来维护商品、订单、客户和商品分类。

知识点清单

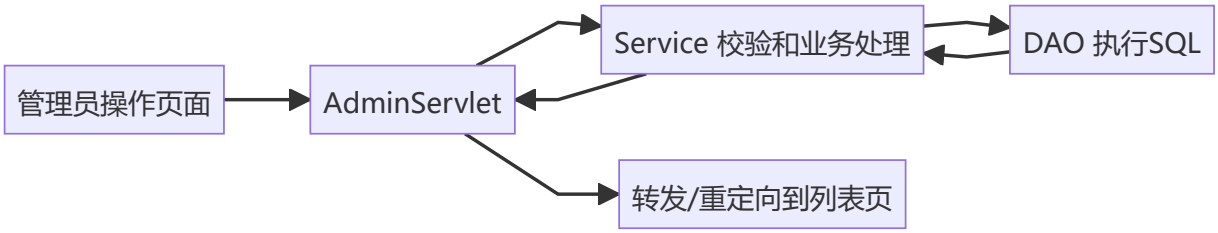
- 后台主要模块：商品管理、订单管理、客户管理、商品类目管理。
- 商品管理：查询、添加、修改、删除、上传图片、加入/移出推荐。
- 订单管理：查询订单列表、查看订单详情、修改状态、删除订单。
- 客户管理：添加客户、修改客户、删除客户、重置密码。
- 商品类目管理：添加分类、修改分类、删除分类。
- 后台权限：后台页面通常需要登录检查，可用 Filter 统一拦截。
- 后台 CRUD：本质是 Servlet 调 Service，Service 调 DAO 执行 SQL。

讲解

后台与前台最大的区别是使用者不同。前台面向顾客，重在浏览和购买；后台面向管理员，重在维护数据。后台代码会大量出现 CRUD，也就是增删改查。

商品添加通常还会涉及文件上传，因为商品需要图片。上传后数据库一般不直接存图片二进制，而是存图片路径或文件名。

后台 CRUD 通用流程



商品添加思路

```
1. 后台表单提交商品名称、价格、分类、图片等信息。
2. Servlet 解析普通表单字段和上传文件。
3. 保存图片到服务器 upload 目录。
4. 把商品信息和图片路径封装成 Goods 对象。
5. Service 调用 Dao 插入数据库。
6. 添加成功后重定向到商品列表。
```

易错点

- 删除分类前要考虑分类下是否还有商品。
- 删除商品通常要考虑订单明细是否引用该商品。
- 后台修改数据后常用重定向回列表，避免刷新重复提交。
- 图片上传要处理文件名重复，常用 `UUID + 原文件名`。
- 后台权限检查适合用 Filter，不要每个 Servlet 都写一遍。

附录：上机作业复习清单

从零学习的练习顺序

1. 写一个 HTML 表单，能提交用户名和密码。
2. 写一个 Servlet，能读取表单参数并输出到浏览器。
3. 写登录功能：成功存 Session，失败转发回 JSP。
4. 写 Cookie 上次访问时间或自动登录。
5. 写 JSP + EL 显示 `request/session` 中的数据。
6. 写 JSTL `c:forEach` 循环显示商品列表。
7. 写 JavaBean，并用 `jsp:setProperty property="*"` 接收表单数据。
8. 按 MVC 改造：JSP 只显示，Servlet 控制，JavaBean 封装数据。
9. 写 Filter 统一处理编码或登录检查。
10. 写 JDBC 查询用户表，改成 `PreparedStatement`。
11. 改用 C3P0 + DBUtils 查询用户列表。
12. 写 Ajax 检查用户名是否存在。
13. 写文件上传：上传商品图片。
14. 写文件下载：下载服务器上的图片或文档。
15. 综合做一个小商城模块：商品列表 + 搜索 + 购物车。

做题时的快速判断

- 看到“请求参数”：想到 `request.getParameter()`、EL 的 `param`。

- 看到“一次请求内共享数据”：想到 `request.setAttribute()` + 转发。
- 看到“跨多次请求保存登录状态”：想到 `session`。
- 看到“浏览器端保存少量数据”：想到 `Cookie`。
- 看到“拦截所有请求”：想到 `Filter` 和 `/*`。
- 看到“Web应用启动/关闭”：想到 `ServletContextListener`。
- 看到“JSP中减少Java代码”：想到 EL + JSTL。
- 看到“数据库连接复用”：想到连接池 + `DataSource`。
- 看到“简化JDBC”：想到 DBUtils + `QueryRunner`。
- 看到“局部刷新”：想到 Ajax。
- 看到“文件上传”：想到 `multipart/form-data` + `Commons-FileUpload`。
- 看到“文件下载”：想到 `Content-Disposition: attachment`。

最重要的主线

```
浏览器页面
  → HTTP 请求
  → Filter 可先拦截
  → Servlet 接收请求并控制流程
  → Service/JavaBean 处理业务
  → DAO/JDBC/DBUtils 访问数据库
  → request/session 保存结果
  → JSP + EL/JSTL 展示
  → HTTP 响应回浏览器
```

只要你能把每个知识点放回这条主线里，Java Web 就不会散。