JavaWeb 前 12 章课后习题答案整合

打印版：题目关键词 + 括号内答案 + 简答/编程模板

使用方式：选择题看括号内选项与答案；判断题看 √/×；简答题背关键词；编程题按模板改类名、URL、字段名。个别题按教材口径整理。

章节目录

第 1 章 HTML / CSS / JavaScript 基础

本章题型：填空 6 题；判断 5 题；选择 6 题

填空题：题目与答案

1. HTML 中文译为（超文本标记语言），主要作用是用 HTML 标签描述网页中的文本、图片、声音等内容。
2. 表单由 3 部分组成：表单控件、提示信息和（表单域）。
3. CSS 中文译为（层叠样式表），用于控制网页样式，实现内容与表现分离。
4. （JavaScript）是可直接嵌入 HTML 的脚本语言，用于实现交互和动态效果。
5. <img/> 必须指定（src）属性，用来指定图片路径。
6. 将 JavaScript 代码写在独立 .js 文件中再用 <script> 引入，称为（外部式 / 外部脚本）。

判断题：题目与答案

1. 事件处理程序可在 JavaScript 中调用，也可在 HTML 中调用。（√）
2. 链入式 CSS 能分离 HTML 与 CSS，是常用且实用的方式。（√）
3. JavaScript 函数名不区分大小写。（×，JavaScript 区分大小写）
4. 通过 <tr> 属性可以控制某一个单元格。（×，单元格用 <td>，<tr> 是行）
5. <td> 用于定义单元格，必须嵌套在 <tr></tr> 中。（√）

选择题：题目与答案

1. var x=3,y=2,z=(x+2)/y; alert(z)，结果是（B：2.5）。
2. CSS 设置字号大小的正确写法是（A：{ font-size: 24px; }）。
3. 不属于 <form> 常用属性的是（B：size）。
4. 按教材函数声明格式，正确定义 JavaScript 函数的是（B：function myDemo(a,b){return a*b};）。
5. 设置单元格背景颜色的属性是（B：bgcolor）。
6. txt1='What a very'; txt2='nice day'; txt3=txt1+txt2，结果是（A：What a verynice day）。

简答题：参考答案

如何引入单独 CSS 文件：在 HTML 的 <head> 中使用 <link rel="stylesheet" type="text/css" href="style.css">。考试写出 rel、type、href 三个点即可。

HTML 注释作用：格式为 <!-- 注释内容 -->。注释不会显示在浏览器页面中，主要用于说明代码、临时屏蔽代码、方便维护。

第 2 章 XML / Tomcat / Web 应用基础

本章题型：填空 5 题；判断 5 题；选择 5 题

填空题：题目与答案

1. XML 称为（可扩展标记语言），用于提供数据描述格式。
2. XML 文档有且仅有一个顶层元素，称为文档元素或（根元素）。
3. XML 声明中 standalone 默认值为（no）。
4. DTD 中最常用的属性类型是（CDATA）。
5. Tomcat 端口号在安装目录下的（conf）文件夹中的 server.xml 中配置。

判断题：题目与答案

1. XML 元素不区分大小写。（×，XML 区分大小写）
2. XML 声明必须包含 version，且 version 应放在其他属性之前。（√）
3. B/S 中浏览器通常不直接连接数据库，而是通过 Web 服务器访问数据库。（√）
4. Tomcat 默认端口号是 8080。（√）
5. webapps 是 Tomcat 发布 Web 应用的主要目录。（√）

选择题：题目与答案

1. 修改 Tomcat 端口要改（B：tomcat/conf/server.xml）。
2. startup.bat 位于（A：bin）目录。
3. Path 中配置 JDK bin 的正确写法是（B：%JAVA_HOME%\bin;）。
4. 正确 XML 文档声明是（A：<?xml version="1.0" encoding="GBK"?>）。
5. XML 描述正确的是（B：所有 XML 元素都必须正确关闭）。

简答题：参考答案

Tomcat 常见目录：bin：启动/关闭脚本；conf：配置文件；lib：公共 jar；logs：日志；temp：临时文件；webapps：发布应用；work：JSP 编译后的工作目录。

IDEA 修改访问路径：Run/Debug Configurations -> Tomcat Server -> Deployment -> Application context，修改为 /项目名 或 / 后重新运行。

第 3 章 HTTP 协议 / Servlet 入门

本章题型：填空 5 题；判断 5 题；选择 8 题

填空题：题目与答案

1. HTTP 即（超文本传输协议），是一种请求/响应式协议。
2. HTTP 请求消息由请求行、（请求头）和实体内容组成。
3. 请求行由（请求方式）、资源路径和 HTTP 协议版本组成。
4. HTTP 定义浏览器与（Web 服务器）之间交换数据的过程和数据格式。
5. Accept-Encoding 常见编码格式为 gzip 和（compress）。

判断题：题目与答案

1. HTTP 消息以明文为主，消息内容可见。（√，HTTPS 除外）
2. POST 比 GET 更安全。（√，教材口径：参数不直接显示在地址栏）
3. 请求方式只有 GET、POST 两种。（×，还有 PUT、DELETE 等）
4. 响应状态行包括 HTTP 版本、状态码、状态描述。（√）
5. Location 头字段可通知客户端请求文档的新地址。（√）

选择题：题目与答案

1. 使资源以下载窗口打开的响应头是（A：Content-Disposition）。
2. Refresh 自动刷新的时间单位是（C：秒）。
3. 通知客户端请求文档新地址的头字段是（D：Location）。
4. 表示服务器错误的状态码是（D：500）。
5. 实现浏览器重定向的状态码是（C：302）。
6. 重写 doGet/doPost 的 Servlet 父类通常是（D：HttpServlet）。
7. LoginServlet extends ____ 应填（B：HttpServlet）。
8. 地址栏直接访问 Servlet 默认发 GET 请求，控制台输出（A：get）。

简答题：参考答案

GET 与 POST 区别：GET 参数拼在 URL 后，长度受限、可见、常用于查询；POST 参数放在请求体中，容量较大、相对安全、常用于提交表单。

常见状态码：200 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。

编程 / 综合题：可套模板

模板：Servlet 基本骨架

```
@WebServlet("/DemoServlet")
public class DemoServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req,
            HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("text/html;charset=utf-8");
        resp.getWriter().write("hello servlet");
    }
    protected void doPost(HttpServletRequest req,
            HttpServletResponse resp) throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

第 4 章 Servlet 请求转发 / 响应输出

本章题型：填空 6 题；判断 5 题；选择 2 题

填空题：题目与答案

1. forward() 将当前请求交给其他 Web 资源处理，这种方式称为（请求转发）。
2. 封装 HTTP 响应消息的接口是（HttpServletResponse）。
3. 服务器让客户端重新发送请求到新资源路径，称为（请求重定向 / 重定向）。
4. 自定义 Servlet 可继承（GenericServlet）或 HttpServlet。
5. Servlet 配置可用 web.xml 或（@WebServlet）注解。
6. Servlet 生命周期三阶段：初始化、运行、（销毁）阶段。

判断题：题目与答案

1. response.getOutputStream() 和 getWriter() 可同时发送响应体。（×，二选一）
2. Tomcat 初始化 Servlet 时会把配置信息封装为 ServletConfig。（√）
3. @WebServlet 的 value 或 urlPatterns 通常必需，且二者不能同时使用。（√）
4. destroy() 在整个生命周期会被调用多次。（×，通常只调用一次）
5. ServletRequest.setAttribute(name,obj) 可把对象保存到 request 域。（√）

选择题：题目与答案

1. HttpServlet 中处理 POST 请求的方法是（C：doPost）。
2. Web 应用中所有 Servlet 共享数据的对象是（B：ServletContext）。

简答题：参考答案

HttpServletResponse 常见状态码：200：成功；302：重定向；404：资源不存在；500：服务器错误。可用 response.setStatus(code) 设置。

请求转发 vs 重定向：转发：服务器内部完成、一次请求、URL 不变、request 域数据保留、可访问 WEB-INF。重定向：客户端重新请求、两次请求、URL 改变、request 域丢失、可跨站点。

编程 / 综合题：可套模板

模板：ChineseServlet

```
@WebServlet("/ChineseServlet")
public class ChineseServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req,
            HttpServletResponse resp) throws IOException {
        resp.setContentType("text/html;charset=utf-8");
        resp.getWriter().write("JavaWeb 程序设计任务教程");
    }
}
```

模板：ShowTimesServlet

```
@WebServlet("/ShowTimesServlet")
public class ShowTimesServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req,
            HttpServletResponse resp) throws IOException {
        ServletContext ctx = getServletContext();
        Integer times = (Integer) ctx.getAttribute("times");
        times = times == null ? 1 : times + 1;
        ctx.setAttribute("times", times);
        resp.setContentType("text/html;charset=utf-8");
        resp.getWriter().write("网站被访问了" + times + "次");
    }
    protected void doPost(HttpServletRequest req,
            HttpServletResponse resp) throws IOException {
        doGet(req, resp);
    }
}
```

第 5 章 Cookie / Session / 购物车

本章题型：填空 5 题；判断 5 题；选择 5 题

填空题：题目与答案

1. 服务器发送 Cookie 时会在响应头增加（Set-Cookie）字段。
2. Web 会话指浏览器与（Web 服务器）之间连续发生的一系列请求和响应。
3. Session 比 Cookie 安全，因为关键数据保存在（服务器）端。
4. Servlet 中保存会话数据的对象是（Cookie）和 Session。
5. Tomcat 会话有效时间可在（web.xml）中设置，默认 30 分钟。

判断题：题目与答案

1. session-timeout 设置为 0 或负数表示会话永不超时。（√）
2. Session 数据保存在服务器端，常借助 Cookie 保存 JSESSIONID。（√）
3. Cookie 的 domain 属性严格区分大小写。（×，域名通常不区分大小写）
4. Cookie path 设置后只对当前访问路径所属目录有效。（×，path 决定匹配范围，可包含子路径）
5. 一次会话只允许一次请求与响应。（×，一次会话可包含多次请求/响应）

选择题：题目与答案

1. setMaxAge(-1) 表示（C：浏览器关闭时 Cookie 被删除）。
2. Tomcat 默认会话超时时间是（B：30 分钟）。
3. 强制 Session 无效的方法是（D：session.invalidate()）。
4. 存在 Session 直接返回，否则返回 null 的方法是（C：request.getSession(false)）。
5. 获取客户端所有 Cookie 的方法是（B：Cookie[] cookies = request.getCookies()）。

简答题：参考答案

什么是会话技术：一个客户端与 Web 服务器之间连续发生的一系列请求和响应。Servlet 中常用 Cookie 与 Session 保存会话数据。

Cookie 与 Session 区别：Cookie 保存在浏览器端，Session 保存在服务器端；Cookie 容量和数量受浏览器限制，Session 较安全；Cookie 依靠 Set-Cookie/Cookie 头，Session 默认依靠名为 JSESSIONID 的 Cookie 找回会话。

编程 / 综合题：可套模板

模板：PurchaseServlet 购物车核心

```
@WebServlet("/PurchaseServlet")
public class PurchaseServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req,
            HttpServletResponse resp) throws IOException {
        String id = req.getParameter("id");
        Book book = BookDB.getBook(id);
        HttpSession session = req.getSession();
        List<Book> cart = (List<Book>) session.getAttribute("cart");
        if (cart == null) {
            cart = new ArrayList<Book>();
            session.setAttribute("cart", cart);
        }
        cart.add(book);
        Cookie c = new Cookie("JSESSIONID", session.getId());
        c.setPath(req.getContextPath());
        c.setMaxAge(60 * 30);
        resp.addCookie(c);
        resp.sendRedirect(req.getContextPath() + "/BookServlet");
    }
}
```

模板：LastAccessServlet 上次访问时间

```
@WebServlet("/LastAccessServlet")
public class LastAccessServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req,
            HttpServletResponse resp) throws IOException {
        resp.setContentType("text/html;charset=utf-8");
        Cookie[] cs = req.getCookies();
        boolean found = false;
        if (cs != null) {
            for (Cookie c : cs) {
                if ("lastAccess".equals(c.getName())) {
                    resp.getWriter().write("上次访问时间：" +
                        URLDecoder.decode(c.getValue(), "utf-8"));
                    found = true;
                }
            }
        }
        String now = URLEncoder.encode(new Date().toString(), "utf-8");
        Cookie nc = new Cookie("lastAccess", now);
        nc.setMaxAge(60 * 60);
        nc.setPath(req.getContextPath());
        resp.addCookie(nc);
        if (!found) resp.getWriter().write("首次访问");
    }
}
```

第 6 章 JSP 基础 / 指令 / 动作标签

本章题型：填空 5 题；判断 5 题；选择 5 题

填空题：题目与答案

1. JSP 中 out 对象用于向（客户端 / 浏览器）发送文本形式的实体内容。
2. JSP 页面处理异常信息可通过（exception）对象实现。
3. 除 RequestDispatcher.forward() 外，还可通过 JSP 的（<jsp:forward>）标签转发。
4. JSP 脚本元素包括 Scriptlets、声明标识和（JSP 表达式）。
5. JSP 第一次访问时先翻译为 Servlet 源文件，再编译为后缀（.class）的文件。

判断题：题目与答案

1. out 隐式对象可直接通过 response.getWriter() 获取。（×，out 是 JspWriter）
2. include 指令 file 属性必须用相对路径，且相对当前文件。（√）
3. include 指令用于静态包含文件，file 只能取相对路径。（√）
4. page 指令所有属性都只能出现一次。（×，import 可多次出现）
5. JSP 文件不能像 HTML 一样直接由浏览器运行，需服务器翻译执行。（√）

选择题：题目与答案

1. out.println('first line') 与 response.getWriter().write('second line') 混用时常先输出（D：second line，再 first line）。
2. 转发动作标签语法正确的是（C：<jsp:forward page="relativeURL"/>）。
3. exception 隐式对象对应类是（B：java.lang.Throwable）。
4. page 指令中告知浏览器文档类型/编码的属性是（B：contentType）。
5. JSP 代码 int x=5; 声明 m(a)=a+1; 后 x=3，结果是（B：x=3,4）。

简答题：参考答案

JSP 九大隐式对象：request 请求；response 响应；session 会话；application 全局上下文；out 输出；pageContext 页面上下文；config 配置；page 当前页面对象；exception 异常对象。

编程 / 综合题：可套模板

模板：b.jsp 静态包含 time.jsp

```
<%@ page contentType="text/html;charset=utf-8" %>
欢迎访问，现在的时间是：
<%@ include file="time.jsp" %>
```

模板：b.jsp 等 5 秒后动态包含 a.jsp

```
<%@ page contentType="text/html;charset=utf-8" %>
b.jsp 页面内容<br>
<% Thread.sleep(5000); %>
<jsp:include page="a.jsp" />
```

第 7 章 EL / JSTL

本章题型：填空 5 题；判断 5 题；选择 5 题

填空题：题目与答案

1. JSTL 由（核心标签库）、国际化/格式化、XML、函数、SQL 标签库组成。
2. EL 的（cookie）隐式对象用于获取客户端 Cookie 信息。
3. EL 的（applicationScope）代表 application 域属性的 Map 对象。
4. <c:forEach> 可迭代 Set、List、Map 和（数组）中的元素。
5. （Core / 核心标签库）包含 Web 应用中通用操作标签。

判断题：题目与答案

1. EL 表达式格式为 ${表达式}。（√）
2. 使用 JSTL 前要用 <%@ taglib %> 引入标签库和前缀。（√）
3. Core 是 JSTL 核心标签库。（√）
4. EL 与 JSP 隐式对象除 pageContext 共有外，其他对象含义不同。（√）
5. EL 条件运算符类似 Java 的 if-else。（√）

选择题：题目与答案

1. <c:else> 标签不存在，代码会编译/运行出错（B：正确写法用 c:choose/c:otherwise）。
2. <c:out value="username1" default="unknown"/> 输出 username1；带 body unknown 但有 value=username2 输出 username2，故（B）。
3. 获取请求头某个值的 EL 隐式对象是（A：header）。
4. ${ (1==2) ? 3 : 4 } 的结果是（D：4）。
5. request.getAttribute("p") 等效 EL 是（D：${requestScope.p}）。

简答题：参考答案

EL 11 个隐式对象：pageContext；pageScope；requestScope；sessionScope；applicationScope；param；paramValues；header；headerValues；cookie；initParam。

编程 / 综合题：可套模板

模板：登录后用 EL 显示用户名密码

```
<!-- LoginServlet 成功后： -->
request.setAttribute("username", "itcast");
request.setAttribute("password", "123");
request.getRequestDispatcher("success.jsp").forward(request, response);

<!-- success.jsp： -->
用户名：${requestScope.username}<br>
密码：${requestScope.password}
```

模板：c_choose.jsp 多条件

```
<c:choose>
  <c:when test="${empty param.username}">unknown user</c:when>
  <c:when test="${param.username == 'itcast'}">itcast is a manager</c:when>
  <c:otherwise>you are a manager</c:otherwise>
</c:choose>
```

模板：提交后显示文本框内容

```
<c:if test="${empty param.msg}">
  <form action="c_out.jsp" method="post">
    <input name="msg"><input type="submit" value="提交">
  </form>
</c:if>
<c:if test="${not empty param.msg}">${param.msg}</c:if>
```

第 8 章 JavaBean / JSP Model / MVC

本章题型：填空 5 题；判断 5 题；选择 5 题

填空题：题目与答案

1. JSP 有两种开发模型：JSP Model1 和（JSP Model2）。
2. JSP Model1 采用（JSP + JavaBean）技术。
3. Model2/MVC 中控制器由（Servlet）实现，视图由 JSP 实现，模型由 JavaBean 实现。
4. MVC 三部分：模型 Model、视图 View、（控制器 Controller）。
5. MVC 中负责处理用户交互的是（控制器 Controller）。

判断题：题目与答案

1. JSP Model1 采用 JSP+Servlet+JavaBean，实际就是 MVC。（×，这是 Model2 的特点）
2. JavaBean 需要默认无参构造方法。（√）
3. Model2 中 Servlet 是控制器，JSP 是视图。（√）
4. JavaBean 属性通常应私有化。（√）
5. 控制器负责管理业务数据和业务规则。（×，这是模型 Model 的职责）

选择题：题目与答案

1. JSP Model2 中实现视图的是（A：JSP）。
2. MVC 中负责展示数据并与用户交互的是（C：视图 View）。
3. Book 类有 private price 与 get/set，且无显式构造时默认有无参构造，故（A：符合 JavaBean 规范）。
4. MVC 特点描述错误的是（B：使耦合性增强）。
5. JSP Model2 中接收浏览器请求的是（C：Servlet）。

简答题：参考答案

JavaBean 规范：public 类；属性 private；提供 public 无参构造；提供符合 getXxx/setXxx/isXxx 命名的访问器；最好实现 Serializable。

编程 / 综合题：可套模板

模板：Student + stuInfo.jsp

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

```
    public void setAge(int age) { this.age = age; }
}
```

```
<jsp:useBean id="stu" class="bean.Student" scope="page"/>
<jsp:setProperty name="stu" property="name" value="张三"/>
<jsp:setProperty name="stu" property="age" value="20"/>
姓名：<jsp:getProperty name="stu" property="name"/>
年龄：<jsp:getProperty name="stu" property="age"/>
```

模板：User 表单自动封装

```
<jsp:useBean id="user" class="bean.User" scope="request"/>
<jsp:setProperty name="user" property="*"/>
用户名：<jsp:getProperty name="user" property="username"/>
密码：<jsp:getProperty name="user" property="password"/>
```

第 9 章 过滤器 / 监听器 / 文件上传下载

本章题型：填空 5 题；判断 5 题；选择 5 题

填空题：题目与答案

1. Filter 中包含（Filter）接口、FilterConfig 接口、FilterChain 接口。
2. 监听 HttpSession 生命周期的接口是（HttpSessionListener）。
3. 文件下载常指定响应头（Content-Type）和 Content-Disposition。
4. ServletFileUpload.parseRequest() 将表单数据封装成（FileItem）列表返回。
5. 文件上传 form 的 enctype 应为（multipart/form-data）。

判断题：题目与答案

1. ServletRequestAttributeListener 可监听 request 属性变更。（√）
2. Filter 位于客户端和处理程序之间，可拦截请求/响应。（√）
3. 上传文件名前添加 UUID 可避免重名覆盖。（√）
4. Filter 链请求拦截顺序与响应拦截顺序相同。（×，响应通常反向返回）
5. 普通表单字段调用 getContentType() 会抛异常。（×，通常返回 null；文本用 getString()）

选择题：题目与答案

1. 不属于 dispatcherTypes 可选值的是（C：RESPONSE）。
2. web.xml 中 <session-timeout>2</session-timeout> 表示空闲（B：2 分钟）后销毁 session。
3. 返回 Filter 所有初始化参数名称的是（D：getInitParameterNames()）。
4. 用于创建/封装上传 FileItem 的工厂类是（C：DiskFileItemFactory）。
5. 获取上传文件字段中文件名的方法是（A：getName()）。

简答题：参考答案

Filter 生命周期：服务器创建 Filter 后调用 init() 一次；每次匹配请求调用 doFilter()；放行写 chain.doFilter(request,response)；应用卸载前调用 destroy() 一次。

编程 / 综合题：可套模板

模板：MyFilter 拦截 MyServlet

```
@WebFilter("/MyServlet")
public class MyFilter implements Filter {
    public void doFilter(ServletRequest request,
            ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
        response.setContentType("text/html;charset=utf-8");
        response.getWriter().write("Hello MyFilter<br>");
        chain.doFilter(request, response);
    }
}
```

模板：DownloadServlet 下载核心

```
@WebServlet("/DownloadServlet")
public class DownloadServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req,
            HttpServletResponse resp) throws IOException {
        String filename = req.getParameter("filename");
        String path = getServletContext().getRealPath("/download/" + filename);
        resp.setContentType(getServletContext().getMimeType(filename));
        resp.setHeader("Content-Disposition",
            "attachment;filename=" + URLEncoder.encode(filename, "utf-8"));
        try (InputStream in = new FileInputStream(path);
             OutputStream out = resp.getOutputStream()) {
            byte[] buf = new byte[1024];
            int len;
            while ((len = in.read(buf)) != -1) out.write(buf, 0, len);
        }
    }
}
```

第 10 章 JDBC

本章题型：填空 5 题；判断 5 题；选择 5 题

填空题：题目与答案

1. 表示 Java 程序和数据库连接的接口是（Connection）。
2. ResultSet 游标移动到最后一行的方法是（last()）。
3. PreparedStatement 用于执行（预编译 / 参数化）的 SQL 语句。
4. executeUpdate(String sql) 可执行 insert、（update）和 delete。
5. 加载 JDBC 驱动并创建连接的类是（DriverManager）。

判断题：题目与答案

1. Statement 对相同 SQL 只编译执行一次。（×，PreparedStatement 才预编译）
2. 使用 DriverManager.registerDriver 可能导致驱动被注册 2 次。（√）
3. ResultSet 表示 select 查询得到的结果集。（√）
4. executeUpdate 返回 int，表示受影响记录数。（√）
5. JDBC 是执行 HTML 语句的 Java API。（×，执行 SQL）

选择题：题目与答案

1. 存储结果集的对象是（A：ResultSet）。
2. 游标从当前位置向下移一行的方法是（A：next()）。
3. executeQuery(String sql) 可执行（C：select 语句）。
4. Statement 中可执行各种 SQL 的方法是（C：execute(String sql)）。
5. 将参数化 SQL 发送到数据库的方法是（B：prepareStatement(String sql)）。

简答题：参考答案

JDBC 六步：1 加载驱动；2 获取 Connection；3 创建 Statement/PreparedStatement；4 执行 SQL；5 处理 ResultSet；6 释放资源。

编程 / 综合题：可套模板

模板：读取 users 表

```
Class.forName("com.mysql.jdbc.Driver");
Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/test", "root", "root");
String sql = "select id,name,password,email from users";
PreparedStatement ps = conn.prepareStatement(sql);
ResultSet rs = ps.executeQuery();
while (rs.next()) {
    System.out.println(rs.getInt("id") + "," +
        rs.getString("name") + "," +
        rs.getString("password") + "," +
        rs.getString("email"));
}
rs.close(); ps.close(); conn.close();
```

模板：PreparedStatement 批处理

```
String sql = "INSERT INTO users(name,password) VALUES(?,?)";
Connection conn = JDBCUtils.getConnection();
PreparedStatement ps = conn.prepareStatement(sql);
for (int i = 1; i <= 5; i++) {
    ps.setString(1, "user" + i);
    ps.setString(2, "pwd" + i);
    ps.addBatch();
}
ps.executeBatch();
JDBCUtils.release(null, ps, conn);
```

第 11 章 C3P0 / DBCP / DBUtils

本章题型：填空 5 题；判断 5 题；选择 5 题

填空题：题目与答案

1. DBUtils 中提供关闭连接、装载驱动等常规操作的类是（DbUtils）。
2. （数据库连接池）负责分配、管理、释放数据库连接，复用已有连接。
3. 单独使用 DBCP 需导入 commons-dbcp.jar 和（commons-pool.jar）。
4. DBCP 可用 BasicDataSource 直接创建，也可通过（BasicDataSourceFactory）创建。

5. C3P0 核心类是（ComboPooledDataSource）。

判断题：题目与答案

1. ResultSetHandler 可把 ResultSet 转换为不同形式。（√）
2. ResultSetHandler 通过 handle(ResultSet rs) 处理结果集。（√）
3. C3P0 的 named-config 可定义零个或多个自定义配置。（√）
4. QueryRunner 与 ResultSetHandler 可简化大部分数据库操作。（√）
5. DBCP 是 Sun 公司开源连接池实现。（×，DBCP 属于 Apache Commons）

选择题：题目与答案

1. QueryRunner 查询指定 id 的单条记录应使用（D：runner.query(sql,new BeanHandler(User.class),new Object[]{id})）。
2. 自定义 ResultSetHandler 应重写（A：handle()）。
3. DbUtils 中装载并注册 JDBC 驱动的方法是（D：loadDriver()）。
4. QueryRunner.query() 可执行（A：SELECT 语句）。
5. DriverManager 与 DataSource 的 close 区别正确的是（B：前者释放连接，后者归还连接池）。

简答题：参考答案

数据库连接池：预先创建一定数量连接，程序使用时从池中取，用完 close() 归还池。优点：减少频繁创建/销毁连接的开销，提高性能，便于统一管理连接数量。

DBUtils 常用类：QueryRunner：query/update/batch；ResultSetHandler：结果集处理；BeanHandler：单个对象；BeanListHandler：对象列表；ScalarHandler：单值。

编程 / 综合题：可套模板

模板：QueryRunner 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), id);

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

第 12 章 Ajax / jQuery

本章题型：填空 5 题；判断 5 题；选择 4 题

填空题：题目与答案

1. 传统异步请求和 Ajax 异步请求都遵循（HTTP）协议。
2. jQuery 的 $.get() 用于向服务器发送（GET）请求。
3. JSON 是一种存储（和交换文本信息 / 键值对）数据的格式。
4. jQuery 选择器用于（查找 / 选取要操作的 HTML 元素）。
5. 浏览器请求包括 GET 请求和（POST）请求。

判断题：题目与答案

1. 传统异步请求常在页面跳转或刷新时发出，每次请求新页面。（√）
2. Ajax 请求过程中页面不发生整体跳转或刷新。（√）
3. jQuery 选择器都以 # 开头。（×，# 只是 id 选择器）
4. jQuery 语法相对 JavaScript 更简单，可提高效率。（√）
5. jQuery 能完全取代 JavaScript。（×，jQuery 是 JavaScript 库）

选择题：题目与答案

1. jQuery 常用操作不包括（C：响应式编程）。
2. jQuery 中最底层 Ajax 方法是（C：$.ajax()）。
3. 用于发送请求的常用方法按原题选（D：$.get()）。
4. Ajax 请求中最基本、最常用的方法之一按原题选（D：$.get()；$.post() 也很常用）。

简答题：参考答案

Ajax 优势：局部刷新、不打断当前页面；异步通信，用户体验好；减少不必要页面传输，节省带宽和服务器压力；可与 JSON 配合进行轻量数据交互。

编程 / 综合题：可套模板

模板：Ajax 检测用户名密码

```
<!-- login.jsp -->
用户名：<input id="username">
密码：<input id="password" type="password">
<button id="btn">登录</button><span id="msg"></span>
<script src="js/jquery.min.js"></script>
<script>
$(function(){
  $("#btn").click(function(){
    $.post("CheckLoginServlet", {
      username: $("#username").val(),
      password: $("#password").val()
    }, function(data){
      $("#msg").html(data);
    });
  });
});
</script>

// CheckLoginServlet doPost
String u = request.getParameter("username");
String p = request.getParameter("password");
response.setContentType("text/html;charset=utf-8");
if ("zhangsan".equals(u) && "123456".equals(p)) {
    response.getWriter().write("输入用户名密码正确，可以使用本系统！");
} else {
    response.getWriter().write("输入用户名或密码错误，请重新输入！");
}
```