

# Java Web 项目知识点检查报告

## 学生信息

- 项目名称：在线图书借阅系统
- 提交时间：2025-10-25 14:30
- 审查工具：Cursor AI + ChatGPT
- 审查时间：2025-10-25 15:00

## 一、整体评估

评估项	得分	满分
知识点覆盖率	85	100
代码规范性	78	100
功能完整性	92	100
安全性	68	100
综合得分	82	100

等级评定：良好（B）

## 二、知识点应用情况详细分析

### ✓ 知识点 1：Servlet 基础与生命周期

应用情况：✓ 已正确应用

检查结果：

- ✓ 正确继承了 `HttpServlet` 类
- ✓ 正确重写了 `doGet()` 和 `doPost()` 方法
- ✓ 使用了 `@webServlet` 注解进行 URL 映射

代码位置：

- `src/servlet/LoginServlet.java:15-45`
- `src/servlet/BookListServlet.java:12-38`

评价：

学生正确理解了 Servlet 的基本概念和使用方法，代码规范。

---

## 知识点 2：请求与响应处理

应用情况：  已正确应用

检查结果：

- ✓ 正确使用 `request.getParameter()` 获取参数
- ✓ 正确设置了响应类型 `response.setContentType()`
- ✓ 处理了中文乱码问题（设置 UTF-8 编码）

代码位置：

- `src/servlet/LoginServlet.java:20-22`
- `src/servlet/BorrowServlet.java:18-20`

示例代码：

```
// LoginServlet.java:20-25
String username = request.getParameter("username");
String password = request.getParameter("password");
response.setContentType("text/html;charset=UTF-8");
```

评价：

正确掌握了请求参数的获取和响应的设置。

---

### ✅ 知识点 3: MVC 模式应用

应用情况: ✅ 已正确应用

检查结果:

- ✓ 正确分离了 Controller (Servlet)、Service、DAO 三层
- ✓ Servlet 只负责请求转发, 业务逻辑在 Service 层
- ✓ 数据库操作封装在 DAO 层
- ✓ JSP 作为视图层, 没有 Java 代码 (使用 JSTL)

代码位置:

- Controller: `src/servlet/` 目录
- Service: `src/service/UserService.java`, `src/service/BookService.java`
- DAO: `src/dao/UserDAO.java`, `src/dao/BookDAO.java`
- View: `webapp/` 目录下的 JSP 文件

架构评价:

MVC 三层架构清晰, 职责分离明确, 代码可维护性好。这是本项目的一大亮点。

---

### ⚠️ 知识点 4: 数据库连接与 JDBC

应用情况: ⚠️ 应用不够规范

检查结果:

- ✓ 使用了 `PreparedStatement` (防止 SQL 注入)
- ✓ 使用 `try-with-resources` 自动关闭资源
- X 问题: 未使用数据库连接池
- ⚠️ 警告: 部分查询未建立索引, 性能有待优化

问题代码位置:

`src/util/DBUtil.java:15-30`

当前代码:

```
// DBUtil.java
public static Connection getConnection() throws SQLException {
    String url = "jdbc:mysql://localhost:3306/library";
    String user = "root";
    String password = "123456";
    return DriverManager.getConnection(url, user, password);
}
```

存在的问题：

1. 每次请求都创建新的数据库连接，效率低下
2. 高并发场景下可能导致连接数耗尽
3. 未使用连接池技术

改进建议：

使用 HikariCP 连接池：

```
// 改进后的 DBUtil.java
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

public class DBUtil {
    private static HikariDataSource dataSource;

    static {
        HikariConfig config = new HikariConfig();
        config.setJdbcUrl("jdbc:mysql://localhost:3306/library");
        config.setUsername("root");
        config.setPassword("123456");
        config.setMaximumPoolSize(10); // 最大连接数
        config.setMinimumIdle(5);      // 最小空闲连接

        dataSource = new HikariDataSource(config);
    }

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

参考资料:

- 教材第 7 章: 数据库连接池
  - HikariCP 官方文档: <https://github.com/brettwooldridge/HikariCP>
- 

## ✗ 知识点 5: Session 管理

应用情况: ✗ 存在明显问题

检查结果:

- ✓ 使用了 Session 存储用户登录状态
- ✗ 严重问题: 未设置 Session 超时时间
- ✗ 问题: 登出功能未调用 `session.invalidate()`
- ⚠ 警告: Session 中存储了过多数据

问题代码位置:

1. `src/servlet/LoginServlet.java:35-40`

问题代码:

```
// LoginServlet.java:35-40
HttpSession session = request.getSession();
session.setAttribute("user", user);
// ✗ 未设置超时时间
response.sendRedirect("index.jsp");
```

2. `src/servlet/LogoutServlet.java:18-22`

问题代码:

```
// LogoutServlet.java:18-22
HttpSession session = request.getSession();
session.removeAttribute("user");
// ✗ 应该使用 session.invalidate() 销毁整个 Session
response.sendRedirect("login.jsp");
```

为什么这是问题？

1. 未设置超时时间：

- 用户关闭浏览器后，Session 仍然存在
- 浪费服务器内存资源
- 可能导致安全问题

2. 登出未销毁 Session：

- `removeAttribute()` 只删除属性，Session 对象仍存在
- 应该使用 `invalidate()` 完全销毁 Session

正确的实现：

```
// 登录时设置超时
HttpSession session = request.getSession();
session.setAttribute("user", user);
session.setMaxInactiveInterval(30 * 60); // 30分钟超时
response.sendRedirect("index.jsp");

// 登出时销毁 Session
HttpSession session = request.getSession(false);
if (session != null) {
    session.invalidate(); // 完全销毁 Session
}
response.sendRedirect("login.jsp");
```


修改优先级：  高（必须修改）

---

## 知识点 6：异常处理与日志

应用情况：  已正确应用

检查结果：

- ✓ 使用了 try-catch 捕获异常
- ✓ 记录了错误日志（使用 `System.out.println`）
-  建议：使用专业的日志框架（如 **Log4j**）

代码位置：

- `src/dao/UserDAO.java:45-58`
- `src/servlet/LoginServlet.java:25-35`

改进建议：

虽然使用了异常处理，但建议使用 Log4j 等专业日志框架：

```
// 推荐使用 Log4j
import org.apache.log4j.Logger;

public class UserDAO {
    private static final Logger logger =
        Logger.getLogger(UserDAO.class);

    public User findByUsername(String username) {
        try {
            // 数据库操作
        } catch (SQLException e) {
            logger.error("查询用户失败: " + username, e);
            throw new RuntimeException("数据库操作失败", e);
        }
    }
}
```

---

## ✗ 知识点 7：安全性考虑

应用情况： ✗ 存在严重安全问题

检查结果：

- ✓ 使用了 `PreparedStatement` 防止 SQL 注入
- ✗ 严重问题：密码明文存储在数据库
- ✗ 严重问题：未对用户输入进行验证
- ✗ 问题：未防止 XSS 攻击

问题 1：密码明文存储

问题代码位置: `src/dao/UserDAO.java:28-35`

```
// UserDAO.java - 注册用户
public void register(User user) throws SQLException {
    String sql = "INSERT INTO users (username, password) VALUES (?, ?)";
    try (Connection conn = DBUtil.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setString(1, user.getUsername());
        ps.setString(2, user.getPassword()); // ✗ 密码明文存储
        ps.executeUpdate();
    }
}
```

为什么这是严重问题？

- 数据库被攻击时，所有用户密码泄露
- 不符合信息安全基本要求
- 违反了《网络安全法》相关规定

正确实现：

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class PasswordUtil {
    /**
     * 使用 SHA-256 加密密码
     */
    public static String encrypt(String password) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            byte[] hash = md.digest(password.getBytes());
            StringBuilder hexString = new StringBuilder();
            for (byte b : hash) {
                String hex = Integer.toHexString(0xff & b);
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);
            }
            return hexString.toString();
        }
    }
}
```



```

        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
}

// 修改后的 UserDao.java
public void register(User user) throws SQLException {
    String sql = "INSERT INTO users (username, password) VALUES (?, ?)";
    try (Connection conn = DBUtil.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setString(1, user.getUsername());
        ps.setString(2, PasswordUtil.encrypt(user.getPassword()));
        // ✅ 加密存储
        ps.executeUpdate();
    }
}

```

修改优先级：🔴 最高（必须立即修改）

问题 2：未验证用户输入

问题代码位置： `src/servlet/LoginServlet.java:20-22`

```

String username = request.getParameter("username");
String password = request.getParameter("password");
// ❌ 未验证输入是否为空、是否符合格式

```

改进建议：

```

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

// ✅ 验证输入
if (username == null || username.trim().isEmpty()) {
    request.setAttribute("error", "用户名不能为空");
    request.getRequestDispatcher("login.jsp").forward(request, response);
    return;
}

```

```
}

if (password == null || password.length() < 6) {
    request.setAttribute("error", "密码长度至少6位");
    request.getRequestDispatcher("login.jsp").forward(request,
response);
    return;
}

// 进行后续处理...
```

修改优先级：  中（建议修改）

---

## 知识点 8: Filter 和 Listener

应用情况：  已正确应用

检查结果：

- ✓ 实现了字符编码过滤器（`CharacterEncodingFilter`）
- ✓ 实现了登录验证过滤器（`LoginCheckFilter`）
- ✓ 正确配置了过滤器的 URL 映射

代码位置：

- `src/filter/CharacterEncodingFilter.java`
- `src/filter/LoginCheckFilter.java`

优秀代码示例：

```
@webFilter("/*")
public class CharacterEncodingFilter implements Filter {
    public void doFilter(ServletRequest request, ServletResponse
response,
                        FilterChain chain) throws IOException,
ServletException {
        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        chain.doFilter(request, response);
    }
}
```

评价：

Filter 的使用规范，体现了对横切关注点的理解。




---

### 三、代码质量分析






代码规范性（78/100）

优点：

-  命名规范，遵循驼峰命名法
-  代码缩进统一
-  大部分类和方法有注释

不足：

-  部分方法缺少 Javadoc 注释
-  存在少量重复代码（如数据库连接获取）
-  魔法数字未定义为常量（如：setMaxInactiveInterval(1800)）

改进建议：

```
// 不好的写法
session.setMaxInactiveInterval(1800);

// 推荐写法
public class Constants {
    public static final int SESSION_TIMEOUT = 30 * 60; // 30分钟
}

session.setMaxInactiveInterval(Constants.SESSION_TIMEOUT);
```

## 安全性分析（68/100）

发现的安全问题：

问题	严重程度	代码位置	状态
密码明文存储	 高	UserDAO.java:32	未修复
未验证用户输入	 中	LoginServlet.java:20	未修复
SQL 注入风险	 已解决	全部使用 PreparedStatement	已解决
XSS 攻击防护	 中	JSP 页面	未处理

必须修复的问题：

1. 密码加密存储（见上文详细说明）
2. 输入验证（见上文详细说明）
3. **XSS 防护**：在 JSP 中使用 `<c:out>` 标签输出用户输入

```
<!-- 不安全的写法 -->
<p>欢迎, <%= username %></p>

<!-- 安全的写法 -->
<p>欢迎, <c:out value="${username}" /></p>
```

## 🧠 界面友好性 (85/100)

优点:

- ✓ 使用了 Bootstrap 美化界面
- ✓ 布局合理，操作流畅
- ✓ 错误提示清晰

建议:

- 可以添加表单验证的前端 JavaScript 提示
- 可以使用 AJAX 实现部分页面无刷新

---

## 四、测试用例通过情况

### 测试结果汇总

测试用例	状态	说明
用户注册功能测试	✅ 通过	正常注册成功
用户登录功能测试	✅ 通过	正确验证用户名密码
SQL 注入防护测试	✅ 通过	成功防止恶意 SQL
Session 管理测试	⚠️ 部分通过	Session 创建正确，但未设置超时
密码加密测试	❌ 失败	密码明文存储
图书借阅功能测试	✅ 通过	借阅逻辑正确
图书归还功能测试	✅ 通过	归还逻辑正确
权限验证测试	✅ 通过	Filter 正确拦截未登录访问





通过率: 75% (6/8)

---





## 五、综合评价与建议

### 总体评价

优点：

1.  架构清晰：MVC 三层架构分离明确，代码组织良好
2.  功能完整：所有必做功能都已实现，运行正常
3.  代码规范：命名规范，有适当的注释
4.  防 SQL 注入：全部使用 PreparedStatement，安全意识较好

不足之处：

1.  密码安全：密码明文存储，这是最严重的问题
2.  性能优化：未使用数据库连接池
3.  Session 管理：未设置超时，登出逻辑不完善
4.  输入验证：缺少完善的输入验证机制

### 改进建议（按优先级排序）

优先级 1（必须修改）：

1. 实现密码加密存储
  - 使用 SHA-256 或 MD5 加密
  - 修改注册和登录逻辑
  - 估计工作量：30 分钟
2. 完善 Session 管理
  - 设置 Session 超时时间
  - 登出时调用 `invalidate()`
  - 估计工作量：15 分钟

优先级 2（强烈建议）：

3. 添加数据库连接池

- 引入 HikariCP 依赖
- 重构 DBUtil 类
- 估计工作量：45 分钟

4. 完善输入验证

- 添加参数验证逻辑
- 前端也添加验证
- 估计工作量：30 分钟

优先级 3（加分项）：

5. 使用 Log4j 日志框架

6. 防止 XSS 攻击

7. 添加单元测试

## 学习建议

1. 复习教材第 9 章：安全性相关内容（密码加密、输入验证）

2. 参考示例项目：教师提供的登录系统标准实现

3. 推荐阅读：

- 《Java Web 安全开发指南》相关章节
- OWASP Top 10 安全风险（中文版）

---

## 六、教师点评

xxx同学：

你的项目整体完成度较好，MVC 架构清晰，代码规范性也不错。特别是 Filter 的应用很规范，体现了你对课程知识点的理解。

但是，密码明文存储是一个严重的安全问题。在实际项目中，这是绝对不允许的。希望你认真对待安全问题，不仅要实现功能，更要确保系统的安全性。

建议学习路径：

1. 先修复密码加密问题（最重要）
2. 然后完善 Session 管理
3. 有余力的话，学习使用连接池技术

期待看到你改进后的代码。相信通过这次实验，你能深刻理解 Java Web 开发中安全性的重要性。

加油！💪

---

审查人： 樊光瑞

审查时间： 2025-10-25 15:00

生成工具： Cursor AI + ChatGPT

下次审查： 修改后重新提交

---

本报告由AI辅助生成，教师已审核确认。