注意，这里截图中，使用用户注册时候的【姓名】进行的搜索，所以有很多相同名字的用户（姓名 可以重复），但是这里显示的名字后面还有灰色字体，这个就是该用户对于的个人空间地址（一定 是唯一的）

前置知识

oracle

```
-- demo表
-- 属性 类型  int id
-- oracle数字类型 number
-- oracle字符类型 varchar2
-- oralce支持的字段都是大写
create table demo(
        id number(8),
        name VARCHAR2(20),
        age number(3)
)

-- 删表
drop table demo2


-- 插入数据(默认不提交 navicat 自动提交)
insert into demo(id,name,age)
values(1,'briup',18);
commit;
-- 主键自增 oracle不支持自增
-- 借助序列自增
select sequence_name from user_sequences
select table_name from user_tables
-- 创建序列
create sequence my_seq

-- 借助哑表 dual 查序列
select my_seq.nextval from dual

insert into demo(name,age)
values('briup',18)

-- 修改数据
update demo set name = 'vanse'
where id = 1

-- 查询数据
select id,name,age from demo;

-- 删除数据
delete from demo
where id = 1;
```

jsp

```
<%--
  Created by IntelliJ IDEA.
  User: vanse
  Date: 2022/6/24
  Time: 11:21
  To change this template use File | Settings | File Templates.
--%>
<%--jsp前缀--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false"
%>
<%--html代码--%>
<html>
<head>
    <title>注册</title>
</head>
<body>
<%--表单--%>
    <%--style每个标签 内置css--%>
<%--
    border: 边框 1px 1个像素
    solid 实线
    black 线的颜色
    width 宽度
    height 高度
--%>
    <div style="border: 1px solid black;width: 208px;height: 80px">
        <%--action 数据提交的位置 java后台--%>
        <%--将来后台代码的路径 register--%>
            <%--name: 将数据提交到后台--%>
            <%--默认提交方式 是get 数据拼接在地址栏后面--%>
            <%--post提交: 数据放在请求体中--%>
        <form action="/register" method="post">
            账号  <input type="text" name="username" placeholder="请输入账
号"><br/>
            密码  <input type="password" name="password" placeholder="请输入
密码"><br>
            <div style="margin-left: 42px;margin-top: 5px;">
                <input type="submit" value="注册">
                <input type="submit" value="登录">
            </div>
        </form>
    </div>
</body>
</html>
```

servlet

概述

Servlet技术可以扩展服务器端的功能，让java代码在服务器端也能成为一种资源（动态资源），客户端浏览器可以通过指定的资源地址（URI），来访问这个java代码（Servlet），同时在Servlet代码中，还可以使用IO流把结果写回给浏览器。

按照Servlet规范要求，编写一个Servlet程序，只需要让自己的类实现`javax.servlet.Servlet`接口就可以了，但是该接口也有一些默认的实现类，所以除了实现接口的方式之外，我们也可以继承它默认的几个实现类，并且重写指定方法，也同样可以完成Servlet程序的编写。

servlet依赖

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
</dependency>
```

实现

LoginServlet

```
package com.briup.demo.web.servlet;

import org.apache.ibatis.session.SqlSession;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/24-06-24-16:56
 * @Description: 登录api
 */
// 需要实现servlet规范
    // implements Servlet
    //HttpServlet 该类已经实现了Servlet
// 该类的访问路径
@WebServlet("/login")
```

```java
public class LoginServlet extends HttpServlet {
    /**
     * 该类的作用: 接收前端数据
     * 表单如果使用了get请求
     * post请求 将来页面请求/resgister 自动调用Post方法
     */
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        doGet(req,resp);
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // soutm soutv
        // HttpServletRequest 封装了请求相关的方法
        // 获取参数 参数乱码
        req.setCharacterEncoding("UTF-8"); // ISO-8859-1 -> UTF-8
        resp.setCharacterEncoding("UTF-8");
        String username = req.getParameter("username");
        System.out.println("username = " + username);
        // 新的编码字符串
        //username = new String(username.getBytes("ISO-8859-1"),"UTF-8");

        String password = req.getParameter("password");
        System.out.println("password = " + password);
        // 获取session范围对象 没有session对象会自动创建session
        HttpSession session = req.getSession();
        // 获取application范围对象 context 泛指上下文
        ServletContext application = req.getServletContext();
        // 页面跳转
            // 如果登录成功(查数据库)  跳转首页
        // 模拟内存数据
        if("briup".equals(username) && "123".equals(password)){
            // 登录成功 跳转首页
            // 请求转发   index.jsp
            // 获取请求转发器 一次请求
            //req.getRequestDispatcher("/index.jsp").forward(req,resp);
            session.setAttribute("username",username);
            // 重定向 (两次请求)
            // 重定向 (两次请求)
            resp.sendRedirect("/index.jsp");

        }else{
            // 登录失败 重新跳转登录页面
            session.setAttribute("msg","账户有误");
//              req.getRequestDispatcher("/login.jsp").forward(req,resp);
            // 重定向
            resp.sendRedirect("/login.jsp");
        }
```

```
    }
}
```

login.jsp

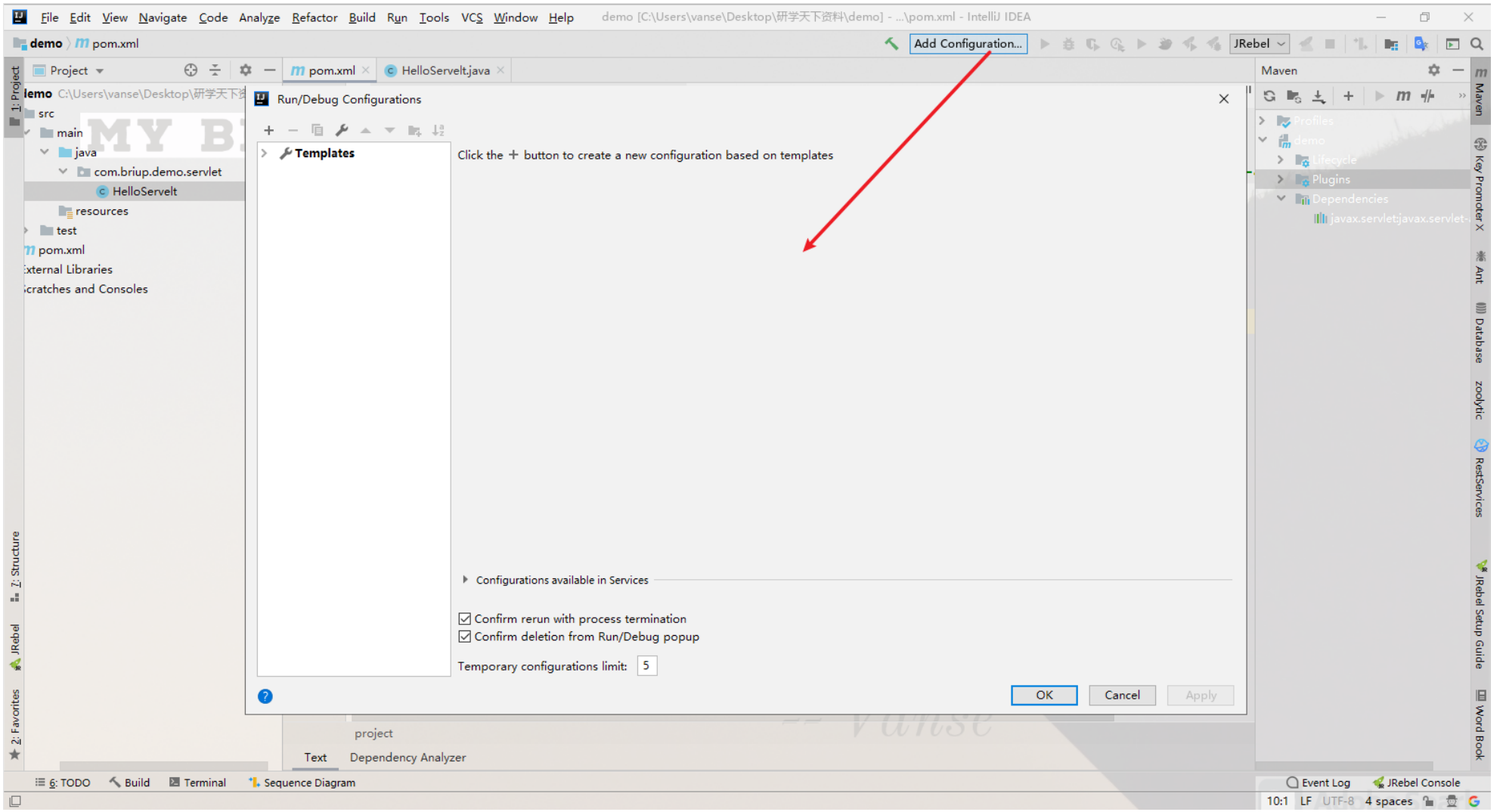
```
<%--
  Created by IntelliJ IDEA.
  User: vanse
  Date: 2022/6/24
  Time: 11:21
  To change this template use File | Settings | File Templates.
--%>
<%--jsp前缀--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false"
%>
<%--html代码--%>
<html>
<head>
    <title>注册</title>
</head>
<script>
    /*js弹框*/
   var msg =  "${msg}";
   // js 条件如果是空 "" undefined
    if(msg){
        // console.log(msg)
        // if条件为真
        window.alert(msg);
    }
    // 把session中的msg移除掉
    <%
        HttpSession session1 = request.getSession();
        session1.removeAttribute("msg");
    %>

</script>
<body>
<%--表单--%>
    <%--style每个标签 内置css--%>
<%--
    border: 边框 1px 1个像素
    solid 实线
    black 线的颜色
    width 宽度
    height 高度
--%>
    <div style="border: 1px solid black;width: 208px;height: 80px">
        <%--action 数据提交的位置 java后台--%>
        <%--将来后台代码的路径 register--%>
            <%--name: 将数据提交到后台--%>
```

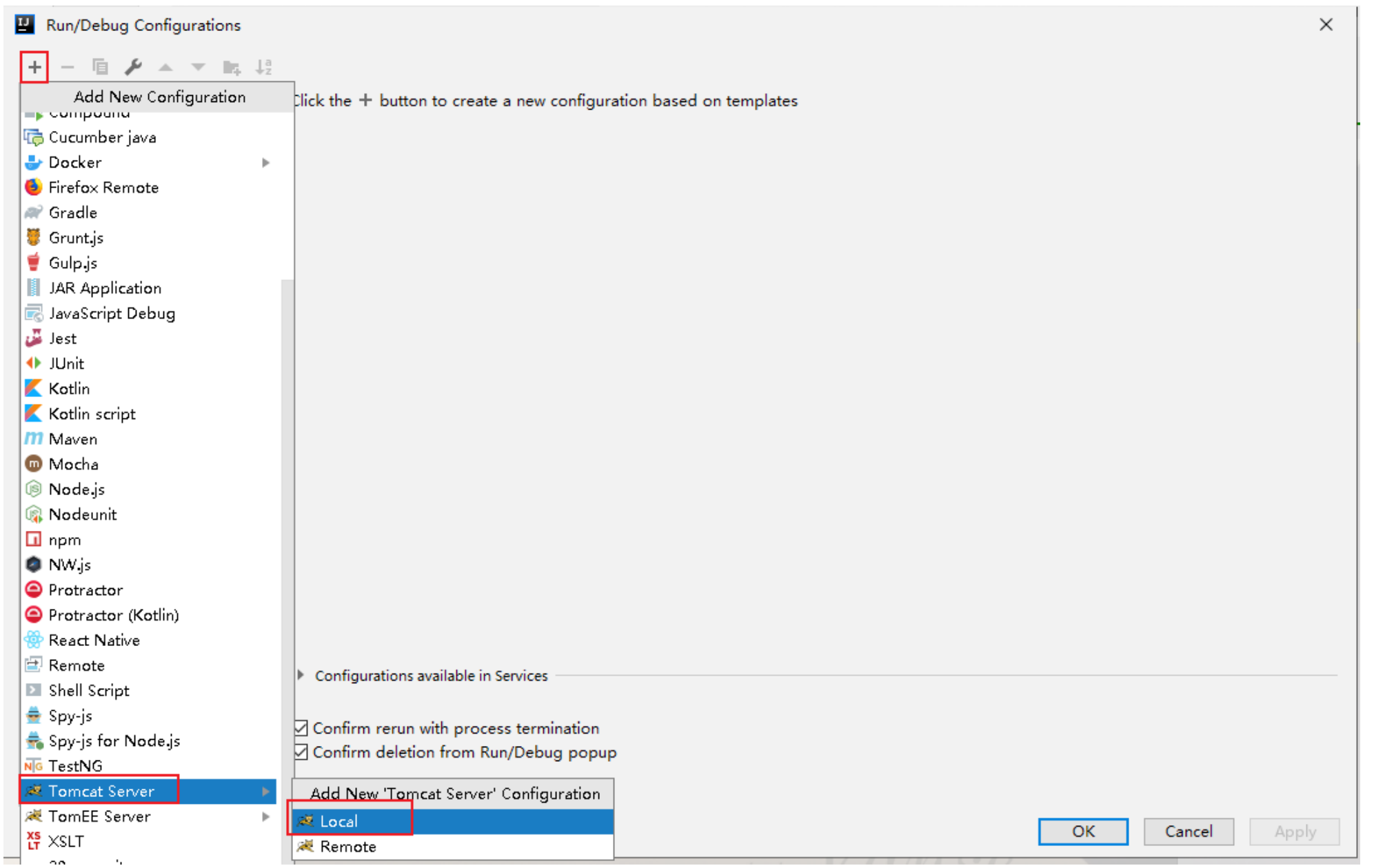
```
            <%--默认提交方式 是get 数据拼接在地址栏后面--%>
            <%--post提交: 数据放在请求体中--%>
        <form action="/login" method="post">
            账号  <input type="text" name="username" placeholder="请输入账
号"><br/>
            密码  <input type="password" name="password" placeholder="请输入
密码"><br>
            <div style="margin-left: 42px;margin-top: 5px;">
                <input type="submit" value="登录">
            </div>
        </form>
    </div>
</body>
</html>
```

index.jsp

```
<%-- isELIgnored="false" 不忽略el表达式 快速取值--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false"
%>
<html>
<body>
<h2>Hello World!</h2>
    欢迎您! ${username}
</body>
</html>
```

集成tomcat


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
demo [C:\Users\vanse\Desktop\研学天下资料\demo] - ...\pom.xml - IntelliJ IDEA
Add Configuration...
JRebel
Project
pom.xml
HelloServelt.java
Maven
Run/Debug Configurations
Templates
Click the + button to create a new configuration based on templates
Configurations available in Services
Confirm rerun with process termination
Confirm deletion from Run/Debug popup
Temporary configurations limit: 5
OK
Cancel
Apply
project
Text
Dependency Analyzer
6: TODO
Build
Terminal
Sequence Diagram
Event Log
JRebel Console
10:1
LF
UTF-8
4 spaces


Run/Debug Configurations
Add New Configuration
Cucumber java
Docker
Firefox Remote
Gradle
Grunt.js
Gulp.js
JAR Application
JavaScript Debug
Jest
JUnit
Kotlin
Kotlin script
Maven
Mocha
Node.js
Nodeunit
npm
NW.js
Protractor
Protractor (Kotlin)
React Native
Remote
Shell Script
Spy-js
Spy-js for Node.js
TestNG
Tomcat Server
TomEE Server
XSLT
Click the + button to create a new configuration based on templates
Configurations available in Services
Confirm rerun with process termination
Confirm deletion from Run/Debug popup
Add New 'Tomcat Server' Configuration
Local
Remote
OK
Cancel
Apply

Run/Debug Configurations

Name: demo_tomcat

Share through VCS

Tomcat Server
demo_tomcat
Templates

Server | Deployment | Logs | Grep Console | Code Coverage | Startup/Connection

Application server: Tomcat 8.5.58 — Configure...

Open browser

After launch: Default ... with JavaScript debugger

URL: http://localhost:8888/

本地tomcat目录

VM options:

On 'Update' action: Restart server — Show dialog

JRE: Default (1.8 - project SDK)

Tomcat Server Settings

HTTP port: 8888

HTTPs port:

JMX port: 1099

AJP port:

Deploy applications configured in Tomcat instance

Preserve sessions across restarts and redeploys

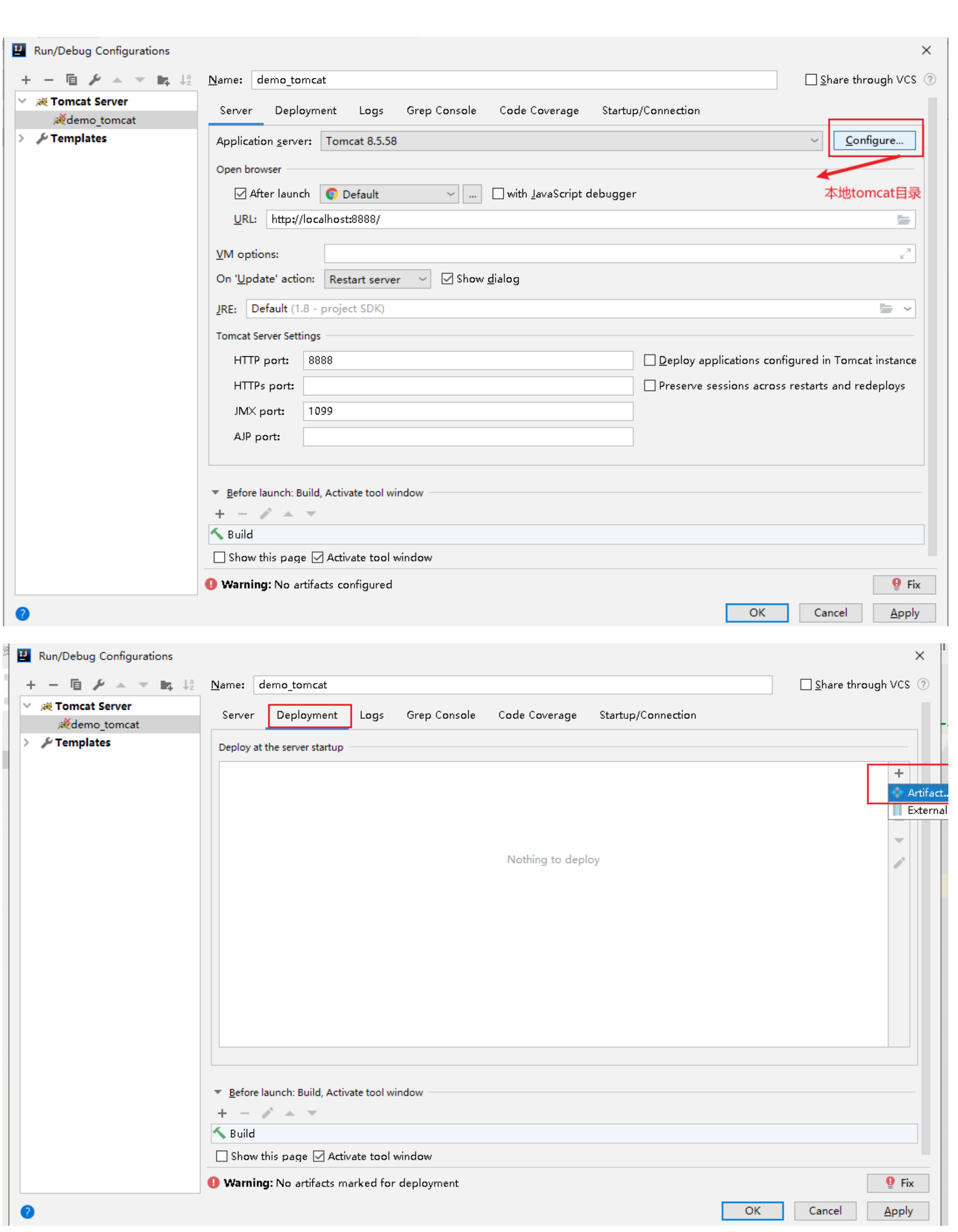
Before launch: Build, Activate tool window

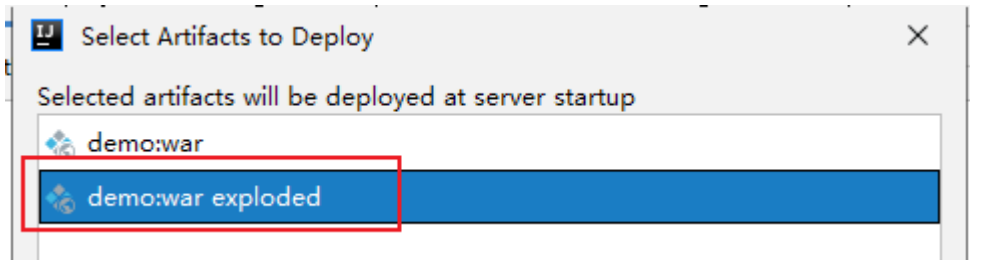
Build

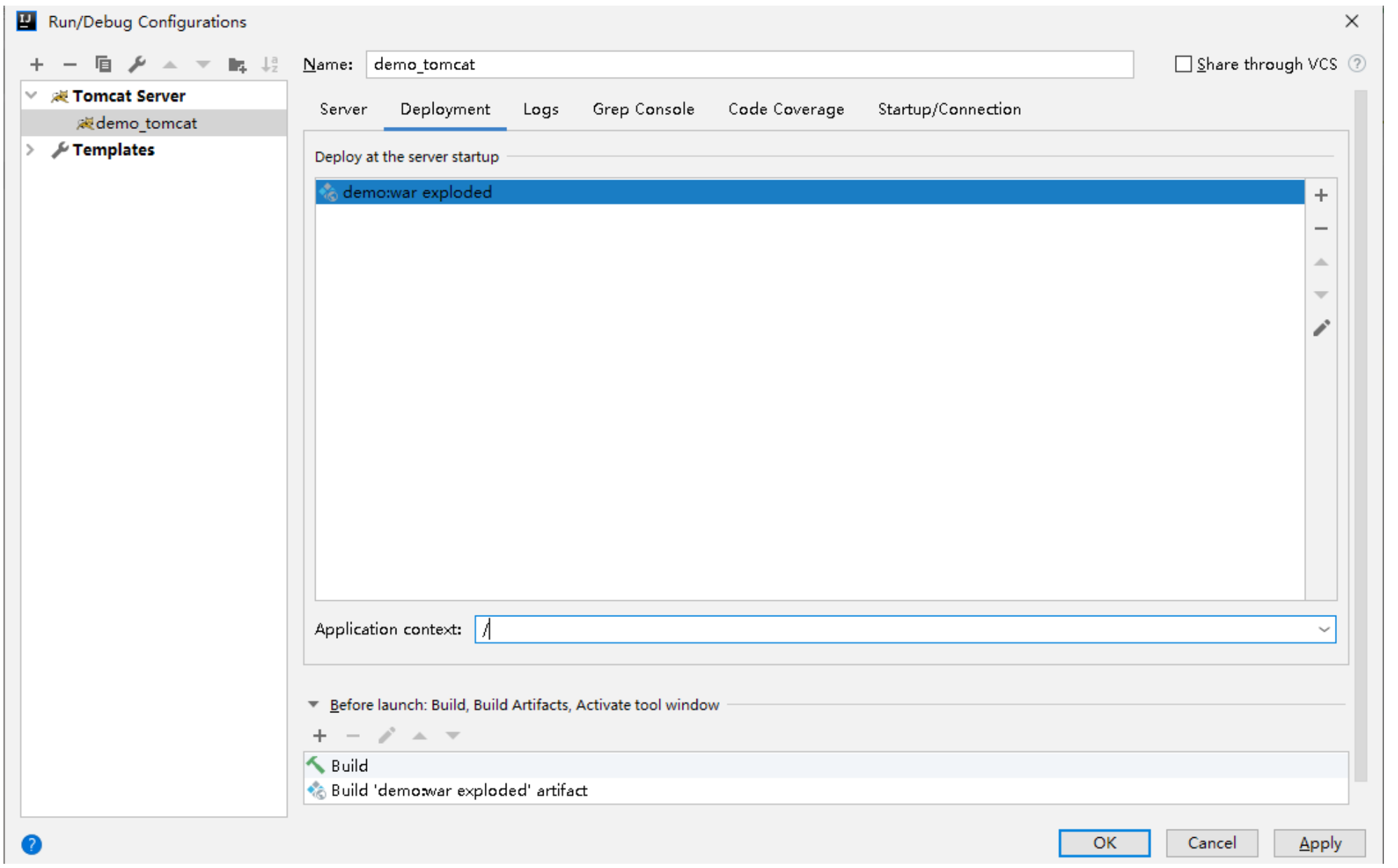
Show this page — Activate tool window

Warning: No artifacts configured — Fix

OK | Cancel | Apply







解决Idea中整合tomcat乱码的问题 启动tomcat时添加vm参数 -Dfile.encoding=UTF-8

mybatis

起源

MyBatis 是一款优秀的持久层开源框架，它支持自定义 SQL、存储过程以及高级映射。MyBatis免除了几乎所有的 JDBC 代码以及设置参数和获取结果集的工作。MyBatis可以通过简单的 XML 或注解来配置和映射原始类型、接口和 Java POJO（Plain Old java Objects，普通老式 Java 对象）为数据库中的记录。

MyBatis的前身是iBATIS，iBATIS于2002年由ClintonBegin创建。MyBatis3是iBATIS的全新设计，支持注解和Mapper。

MyBatis流行的主要原因在于它的简单性和易使用性。在Java应用程序中，数据持久化层涉及到的工作有：

- 将从数据库查询到的数据生成所需要的Java对象
- 将Java对象中的数据通SQL持久化到数据库中

MyBatis通过抽象底层的JDBC代码，自动化封装SQL结果集产生Java对象、Java对象的数据持久化数据库中的过程使得对SQL的使用变得容易。

概述

> **开源的orm映射框架 (封装了jdbc)**

- 依赖 mybaits/驱动/lombok
- 实体/表 orm
- mybatis配置
    - 核心配置文件(数据源,映射文件)
    - 映射器
        - 映射接口(和sql文件绑定)
            - 调用接口方法就能执行sql
        - 映射文件 xml

mybatis四大组件

- SqlSessionFactoryBuilder(构建器模式)(核心配置文件)
- SqlSessionFactory(工厂模式)
- SqlSession(Connection) 主体
- 映射器

实现

依赖

pom.xml

```
<dependencies>
        <!--mybatis-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.5.7</version>
```

```
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.oracle.ojdbc/ojdbc8 -->
        <dependency>
            <groupId>com.oracle.ojdbc</groupId>
            <artifactId>ojdbc8</artifactId>
            <version>19.3.0.0</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <!--lomboc 简化set/get/toString/构造器方法-->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.12</version>
            <scope>provided</scope>
        </dependency>

        <!--测试单元jar-->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
            <scope>test</scope>
            <!--<scope>compile</scope>-->
        </dependency>

    </dependencies>
```

核心配置

mybatis-config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!--外置数据源-->
    <properties resource="db.properties"/>
    <!--数据源-->
    <environments default="development">
        <environment id="development">
        <transactionManager type="JDBC" />
        <dataSource type="POOLED">
            <property name="driver" value="${jdbc.driver}" />
            <property name="url" value="${jdbc.url}" />
            <property name="username" value="${jdbc.username}" />
            <property name="password" value="${jdbc.password}" />
        </dataSource>
    </environment>
```

```
    </environments>

    <!--映射文件位置 将来只需要加载核心配置文件-->
    <mappers>
        <mapper resource="mapper/UserMapper.xml" />
    </mappers>
</configuration>
```

映射接口

UserMapper

```
package com.briup.mybatis.mapper;

import com.briup.mybatis.entity.User;

import java.util.List;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/25-06-25-16:22
 * @Description: 映射接口 -> 和映射文件一致
 */
public interface UserMapper {
    // 接口中的方法要和映射文件的sql一致
    void insertUser(User user); // 插入用户
    void updateUser(User user); // 修改用户
    List<User> query();  // 查询所有用户
    void deleteById(int id); // 根据id删除
    User queryById(int id); // 根据id查询
}
```

映射文件

src/main/resources/mapper/UserMapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.briup.mybatis.mapper.UserMapper">
    <insert id="insertUser" parameterType="com.briup.mybatis.entity.User">

        insert into t_user(id,username,password,age)
        values (my_seq.nextval,#{username},#{password},#{age})
```

```
</insert>

<update id="updateUser" parameterType="com.briup.mybatis.entity.User">
    update t_user set username=#{username},password=#{password}
    where id = #{id}
</update>

<!--根据id删除-->
<delete id="deleteById" parameterType="int">
    delete from t_user
    where id = #{id}
</delete>

<!--
    resultType: 封装的返回类型
    前提: 列名和字段名 使用resultType 才能自动映射
        xml ->  java
        conn.prepareStatement(sql);
-->
<select id="query" resultType="com.briup.mybatis.entity.User">
    select id,username,password,age from t_user
</select>

<!--复杂映射-->
<!--resultmap
    手动映射
    处理多表的关系
-->
    <resultMap id="userMap" type="com.briup.mybatis.entity.User">
        <!--User中的属性 手动映射-->
        <!--id 独立-->
        <id property="id" column="id"/>
        <!--其他属性-->
        <result property="username" column="user_name"/>
        <!--如果列名和属性名一致 可省略-->
        <!--<result property="password" column="password"/>
        <result property="age" column="age"/>-->
    </resultMap>
    <select id="queryById" resultMap="userMap" parameterType="int">
        select id,user_name,password,age from t_user
        where id = #{id}
    </select>
   <!-- <select id="queryById" resultType="com.briup.mybatis.entity.User"
parameterType="int">
        select id,user_name as username,password,age from t_user
        where id = #{id}
    </select>-->

</mapper>
```

加载函数

TestUser

```java
import com.briup.mybatis.entity.User;
import com.briup.mybatis.mapper.UserMapper;
import com.briup.mybatis.utils.SqlSessionUtil;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/25-06-25-17:15
 * @Description: 测试mybatis
 */
public class TestUser {
    @Test
    public void insert(){
        // 使用mybatis 操作数据库
        // 核心配置文件 -> Inputstream
        // TestUser.class.getClassLoader().getResourceAsStream("xml");
        // alt+shift+z
        try {
            InputStream in = Resources.getResourceAsStream("mybatis-config.xml");
            // 构建器对象
            SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
            // 工厂对象
            SqlSessionFactory factory = builder.build(in);
            // SqlSession连接对象
            SqlSession session = factory.openSession();
            // 调用接口中方法
            System.out.println(session);
            UserMapper userMapper = session.getMapper(UserMapper.class);
            User user = new User(1,"briup","000",18);
            userMapper.insertUser(user);
            // 需要手动提交事务
            session.commit();
            // 关闭资源
            session.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```java
@Test
public void update(){
    // alt + shift + m
    try {
        InputStream in = Resources.getResourceAsStream("mybatis-config.xml");
        // 构建器对象
        SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
        // 工厂对象
        SqlSessionFactory factory = builder.build(in);
        // SqlSession连接对象
        SqlSession session = factory.openSession();
        // 调用接口中方法
        UserMapper userMapper = session.getMapper(UserMapper.class);
        User user = new User(1,"vanse","vanse",18);
        userMapper.updateUser(user);
        // 需要手动提交事务
        session.commit();
        // 关闭资源
        session.close();

    } catch (IOException e) {
        e.printStackTrace();
    }
}
@Test
public void update2(){
    SqlSession session = SqlSessionUtil.getSqlSession();

    UserMapper userMapper = session.getMapper(UserMapper.class);
    User user = new User(1,"vanse","123",18);
    userMapper.updateUser(user);

    session.commit();
    session.close();
}

// 查询(查数据 不需要事务)
@Test
public void query(){
    SqlSession session = SqlSessionUtil.getSqlSession();

    UserMapper userMapper = session.getMapper(UserMapper.class);
    // 查询用户
    List<User> userList = userMapper.query();
    // userList.iter
    for (User user : userList) {
        System.out.println(user);
    }
    session.close();
}
```

```java
    // 删除
    @Test
    public void deleteById(){
        SqlSession sqlSession = SqlSessionUtil.getSqlSession();
        UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
        // 代码的红色是报错
        userMapper.deleteById(1);

        sqlSession.commit();
        sqlSession.close();
    }

    // 根据id查询
    @Test
    public void queryById(){
        SqlSession sqlSession = SqlSessionUtil.getSqlSession();
        UserMapper mapper = sqlSession.getMapper(UserMapper.class);
        System.out.println(mapper.queryById(1));

    }
}
```

日志

pom.xml

```xml
<!-- 日志框架log4j的依赖 -->
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

log4j.properties

```
log4j.rootLogger=DEBUG, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d [%-5p] %c - %m%n
#show sql
log4j.logger.java.sql.ResultSet=INFO
log4j.logger.org.apache=INFO
log4j.logger.java.sql.Connection=DEBUG
log4j.logger.java.sql.Statement=DEBUG
log4j.logger.java.sql.PreparedStatement=DEBUG
```

封装

SqlSessionUtil

```java
package com.briup.mybatis.utils;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.IOException;
import java.io.InputStream;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/26-06-26-9:13
 * @Description: SqlSession工具类
 */
public class SqlSessionUtil {
    // 单例模式
    private static SqlSessionFactory factory = null;
    // 获取工厂
    public static SqlSessionFactory getSqlSessionFactory() {

        InputStream in = null;
        try {
            in = Resources.getResourceAsStream("mybatis-config.xml");
            // 获取工厂对象
            if (factory == null) {
                factory = new SqlSessionFactoryBuilder().build(in);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return factory;
    }

    // 静态方法 手动提交
    public static SqlSession getSqlSession() {
        return getSqlSession(false);
    }

    // 给定参数 (true: 自定提交  false: 手动提交)
    public static SqlSession getSqlSession(boolean autocommit) {
        return getSqlSessionFactory().openSession(autocommit);
    }
}
```

三层架构

在web项目的代码结构中，经常会使用到三层架构：

- web层 servlet/filter/listener
- service层
- dao层

web层中代码的主要任务：

- 接收客户端传来的参数
- 把参数封装成对象 saveUser(user)
- 把封装好的对象/数据传给service
- 根据service层的处理结果决定把那个页面/数据返回给客户端

service层中代码的主要任务：

- 接收web层传过来的对象/数据(如果有的话)
- 根据这些信息进行业务逻辑处理

例如，完成一个登录功能，web层接收到用户名和密码之后，把数据传给service层，service层就要根据这些信息来判断用户名是否存在、密码是否正确、用户是否有权限、用户状态当前是否可用、用户是否推送信息、是否给用户相关提示等，这些都属于登录的业务逻辑处理

dao层中代码的主要任务：

- 接受service传的参数(如果有的话)
- 和数据库进行交互
- 把交互结果返回给service层

思考，项目的代码为何要分层编写，全部代码写到一个类中进行实现，是否可以?

web

```java
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("utf-8");
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        UserService userService = new UserServiceImpl();
        User user = userService.login(username, password);
        req.getSession().setAttribute("user",user);
        resp.sendRedirect("index.jsp");
    }
}
```

service

```java
package com.briup.before.service.impl;

import com.briup.before.bean.User;
import com.briup.before.mapper.UserMapper;
import com.briup.before.service.UserService;
import com.sun.deploy.util.StringUtils;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.IOException;
import java.io.InputStream;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/22-06-22-4:54
 * @Description: com.briup.before.service.impl
 */
public class UserServiceImpl implements UserService {
    @Override
    public User login(String username, String password) {
        try {
            InputStream in = Resources.getResourceAsStream("mybatis-config.xml");
            SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(in);
            SqlSession session = factory.openSession();
            UserMapper userMapper = session.getMapper(UserMapper.class);
            User user = userMapper.login(username, password);
            if (user == null) {
                throw new RuntimeException("用户名或者密码有误");
            }
```

```
            return user;
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

dao

```
 User login(@Param("username") String username, @Param("password") String password);


<select id="login" resultType="com.briup.before.bean.User">
          select * from t_user
          where username=#{username} and password=#{password}
</select>
```

其他

- 工具类
- 过滤器
- ...

estore

项目概述

概述

estore系统，也成为网上商城，主要业务是完成在线购物的功能。

estore系统包括前台商家系统、买家系统及后台管理员操作系统，其功能包括：

- 用户浏览
- 购物车
- 商品管理
- 用户管理
- 订单管理
- 配送管理

- 广告推广
- ...

如图：

目标

通过该项目的编写，需要完成的学习目标如下：

- 掌握软件开发流程
- 完成软件开发项目
- 积累软件开发经验
- 巧用软件开发技术
- 熟悉电商业务模型
- 发扬团队合作精神
- 提高语言表达能力
- 锻炼文档书写能力
- 分析解决问题能力

流程

软件开发的流程，大致可以分为：

- 软件需求分析（Software Requirements Analysis）
- 数据建模（ Data Model ）
- 编码/单元测试（Coding）
- 联合调试（Integration）
- 系统测试（Testing）
- 运行维护（Installation & Maintenance）

###模型

常见的软件开发过程模块主要分为：

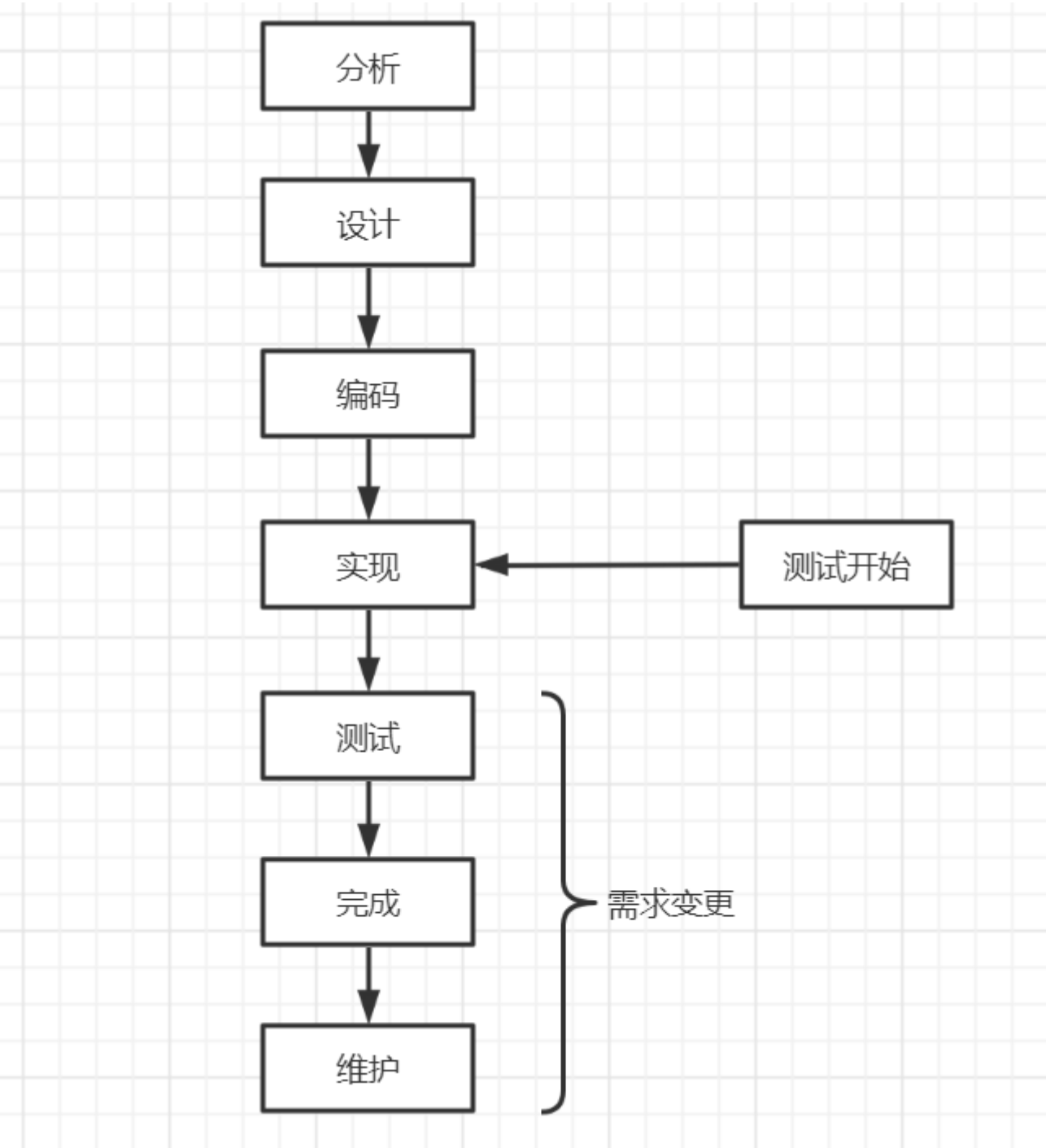
- 瀑布模型
- 快速原型模型
- 螺旋模型

1、瀑布模型

它是线性模型的一种，在所有模型中占有重要地位，是所有其他模型的一个基础

每一个阶段执行一次，按线性顺序进行软件开发。

如图：



在该模型中，测试阶段处于软件实现后，必须在代码完成后留出足够的时间来进行测试，否则将导致测试不充分，并且很多问题到项目后期才暴露。

瀑布模型的优点：

- 开发的各个阶段比较清晰
- 强调早期计划及需求调查
- 适合需求稳定的产品开发

瀑布模型的缺点：

- 依赖于早期的需求调查，不适应需求的变化
- 单一流程不可逆
- 风险往往延至后期才显露，失去及早纠正的机会
- 问题在项目后期才开始暴露
- 前面未发现的错误会传递并扩散到后面的阶段，可能导致项目失败

2、快速原型模型

它是沿用瀑布模型的线性思想，细化了各个阶段，在某些重要关注的阶段之间掺入迭代的思想

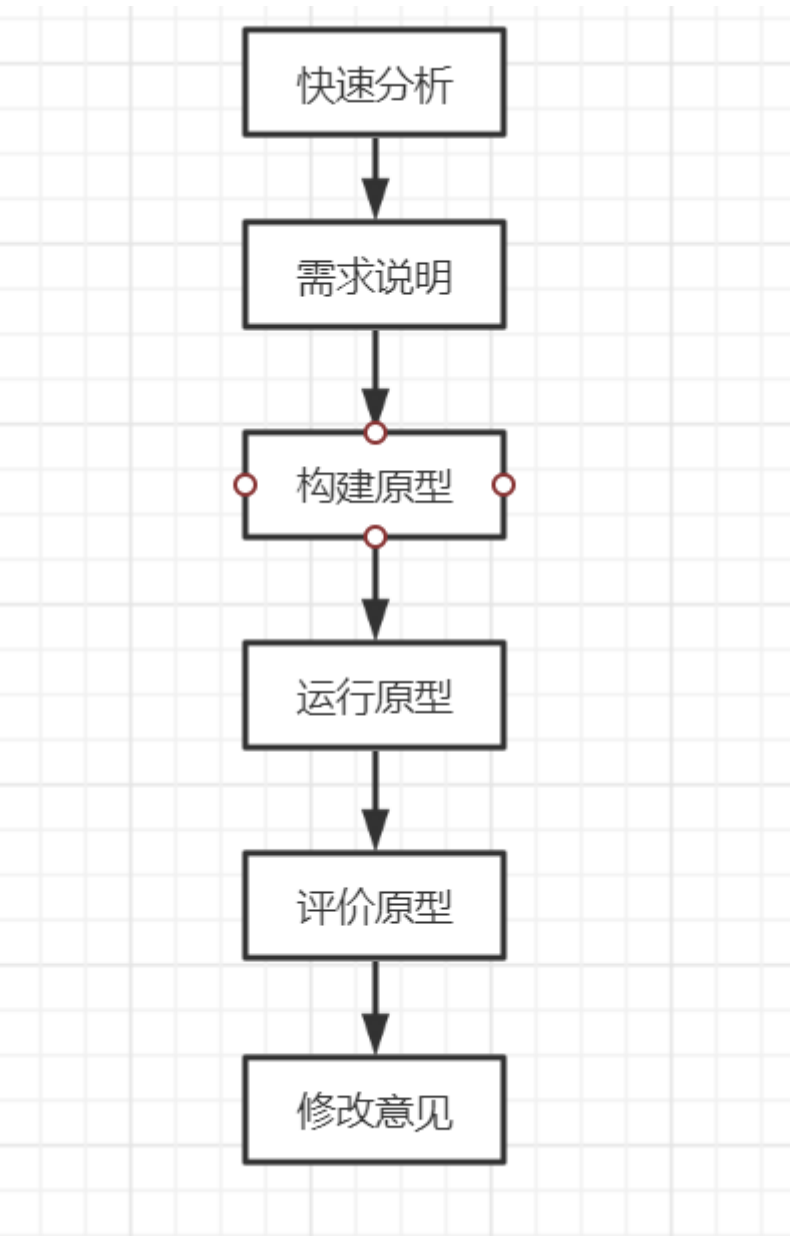
在开发真实系统之前，构造一个原型，在该原型的基础上，逐渐完成整个系统的开发工作。

例如，

第一步，建造一个快速原型，实现用户与系统的交互，用户对原型进行评价，进一步细化软件的需求。通过逐步调整原型使其满足用户的要求，开发人员可以确定用户的真正需求是什么。

第二步，是在第一步的基础上开发出用户满意的软件产品。

如图：



注意，原型中，只是粗糙的对功能进行实现，和用户进行沟通、测试、评价、修改，迭代这个过程，最终获取用户真正想要的软件功能

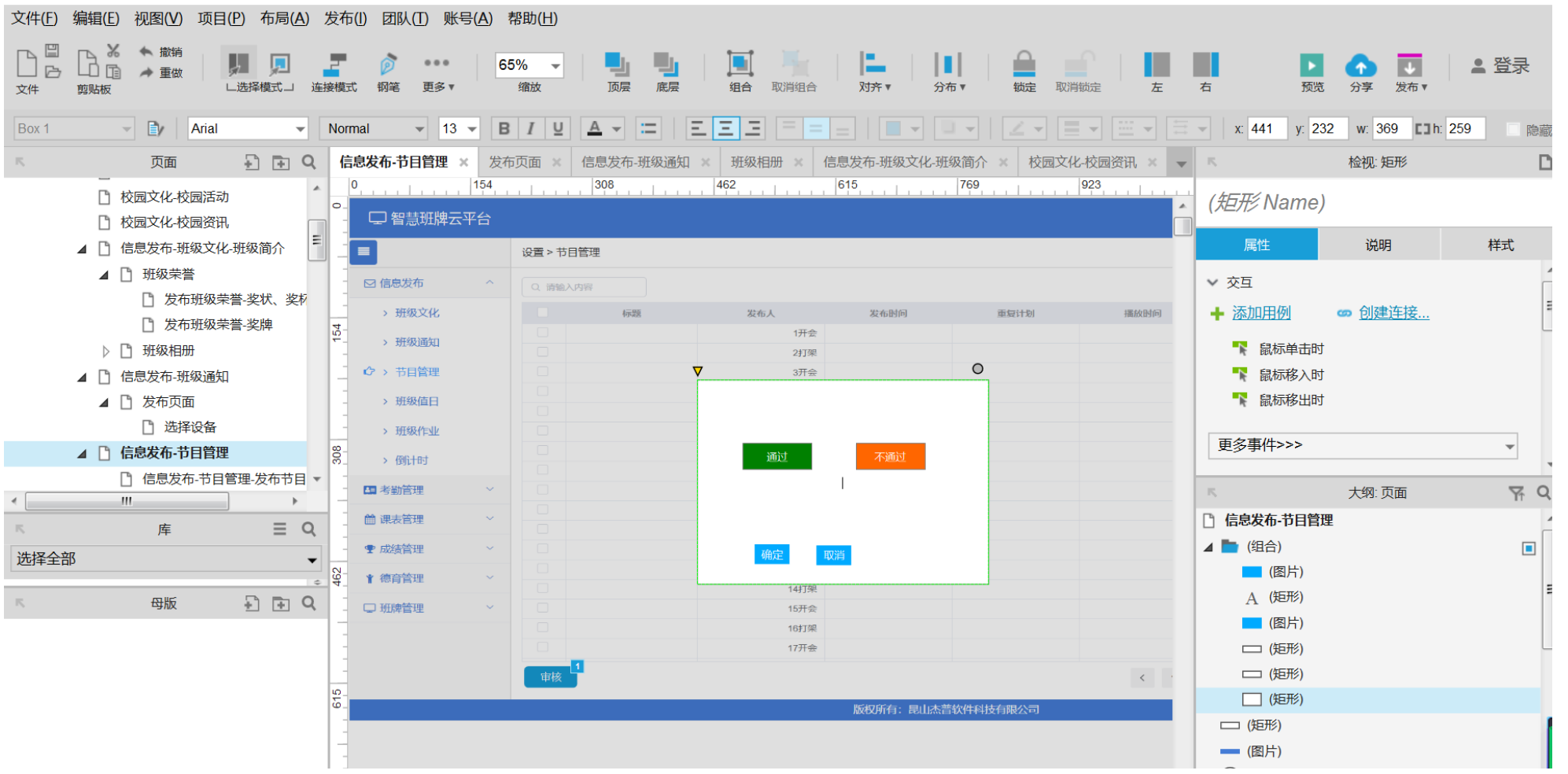
快速原型模型优点：

- 克服瀑布模型的缺点，更好地满足用户的需求并减少由于软件需求不明确带来的项目开发风险。
- 适合预先不能确切定义需求的软件系统的开发。

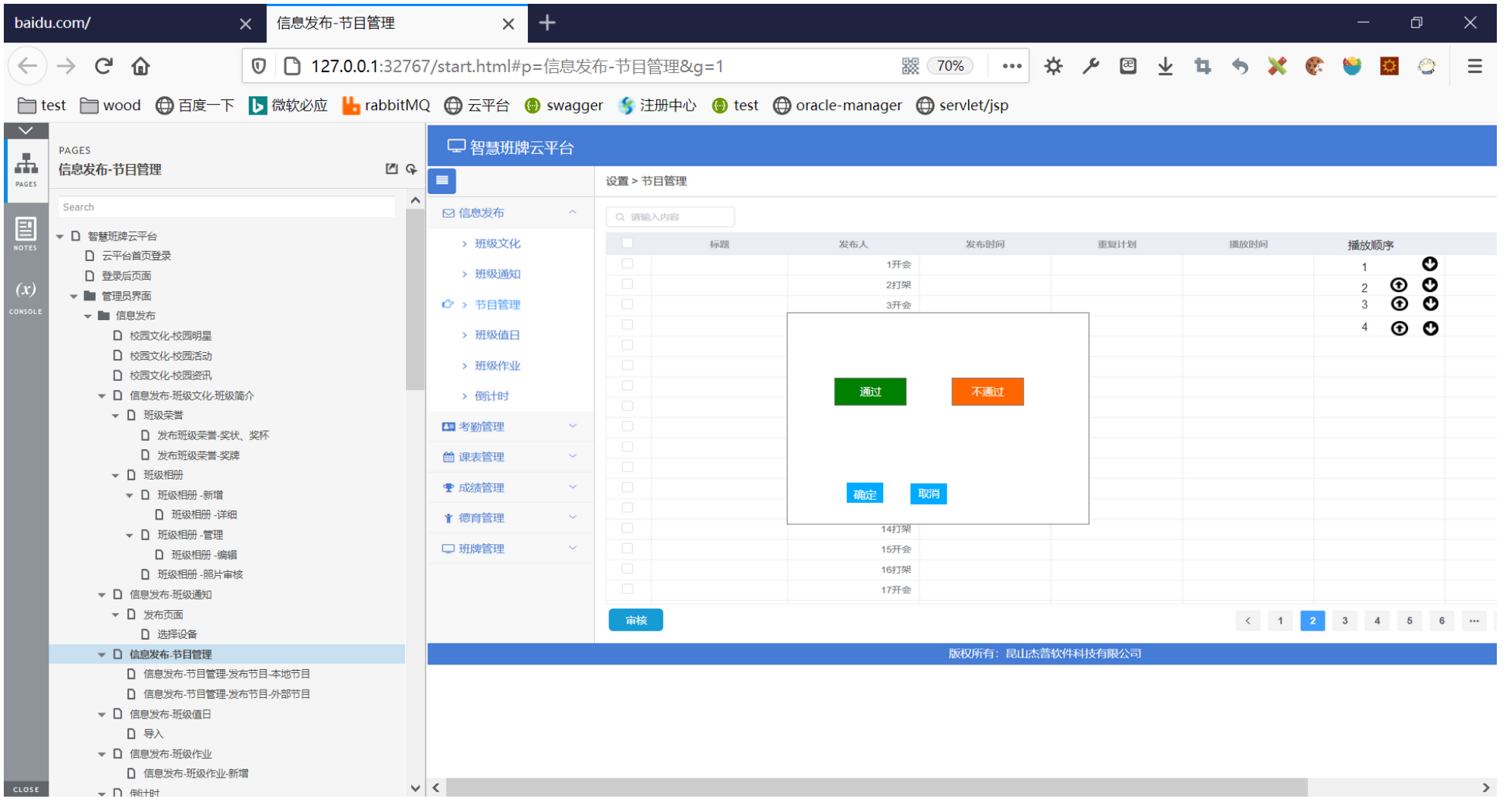
快速原型模型缺点：

- 不适合大型系统的开发，对于小型的、灵活性高的系统较为合适
- 前提要有一个展示性的产品原型，并且需要专门学习相应的工具使用

有很多专门快速创建原型的工具，例如，Axure



构建完成后，可以在浏览器中发布预览：



3、螺旋模型

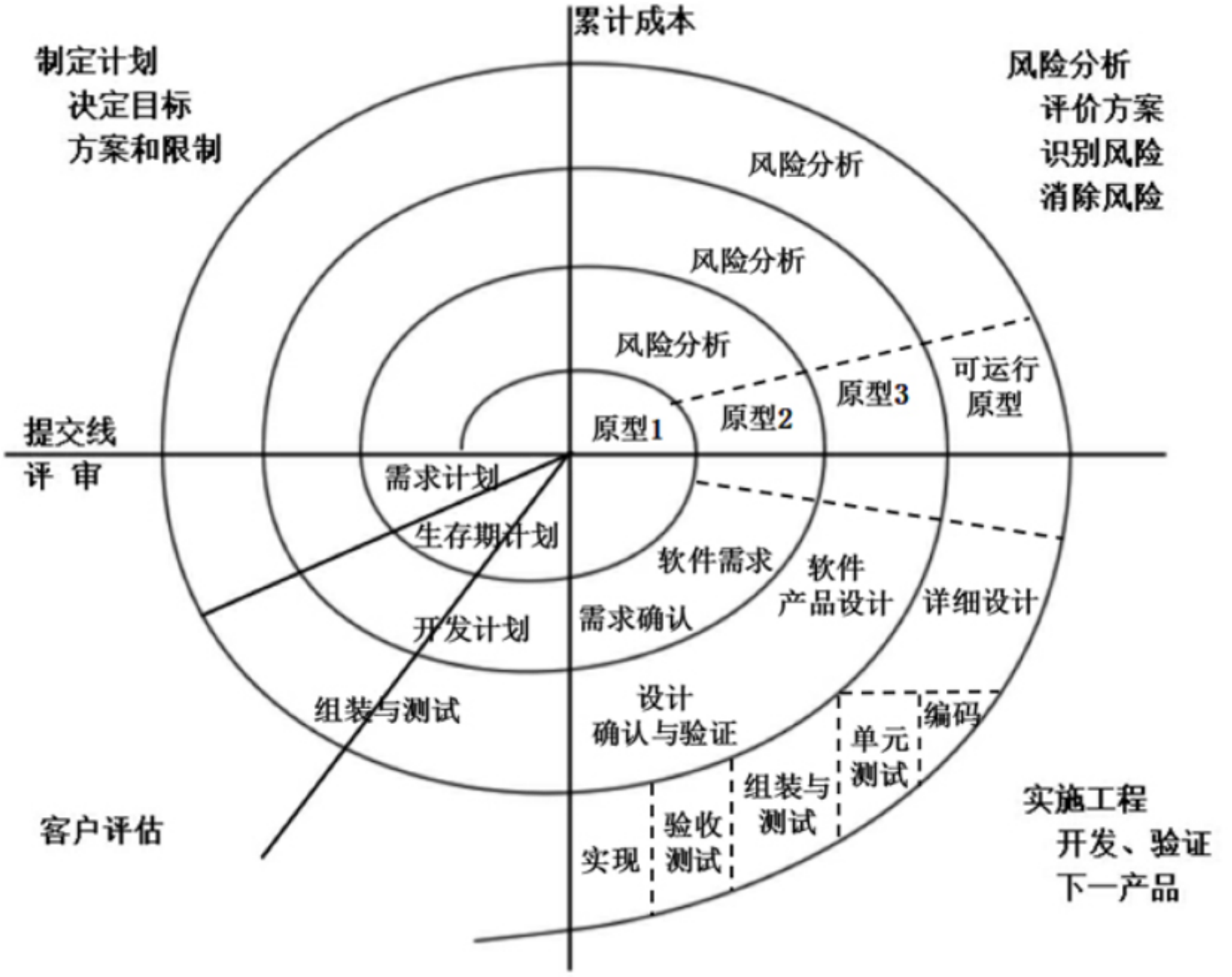
> **螺旋模型是以快速原型模型为基础，在每个项目阶段使用瀑布模型法**

这种模型的每一个周期都包括需求定义、风险分析、工程实现和评审4个阶段，由这4个阶段进行迭代。

螺旋模型即是一种引入了风险分析与规避机制的过程模型，是瀑布模型、快速原型方法和风险分析方法的有机结合。

螺旋模型最大的特点在于引入了其他模型不具备的风险分析，使软件在无法排除重大风险时有机会停止，以减小损失。

如图：



螺旋模型沿着螺线进行若干次迭代，图中的四个象限代表了以下活动：

- 制定计划：确定软件目标，选定实施方案，弄清项目开发的限制条件
- 风险分析：分析评估所选方案，考虑如何识别和消除风险
- 实施工程：实施软件开发和验证
- 客户评估：评价开发工作，提出修正建议，制定下一步计划

每轮循环包含如下六个步骤：

1. 确定目标
2. 识别并化解风险
3. 评估
4. 开发并测试当前阶段
5. 规划下一阶段
6. 确定进入下一阶段的方法步骤

螺旋模型强调**风险分析**，使得开发人员和用户对每个演化层出现的风险有所了解，继而做出应有的反应，因此特别适用于庞大、复杂并具有高风险的系统。

螺旋模型优点：

- 螺旋模型很大程度上是一种风险驱动的方法体系
- 每个阶段之前及经常发生的循环之前，都必须首先进行风险评估

螺旋模型缺点：

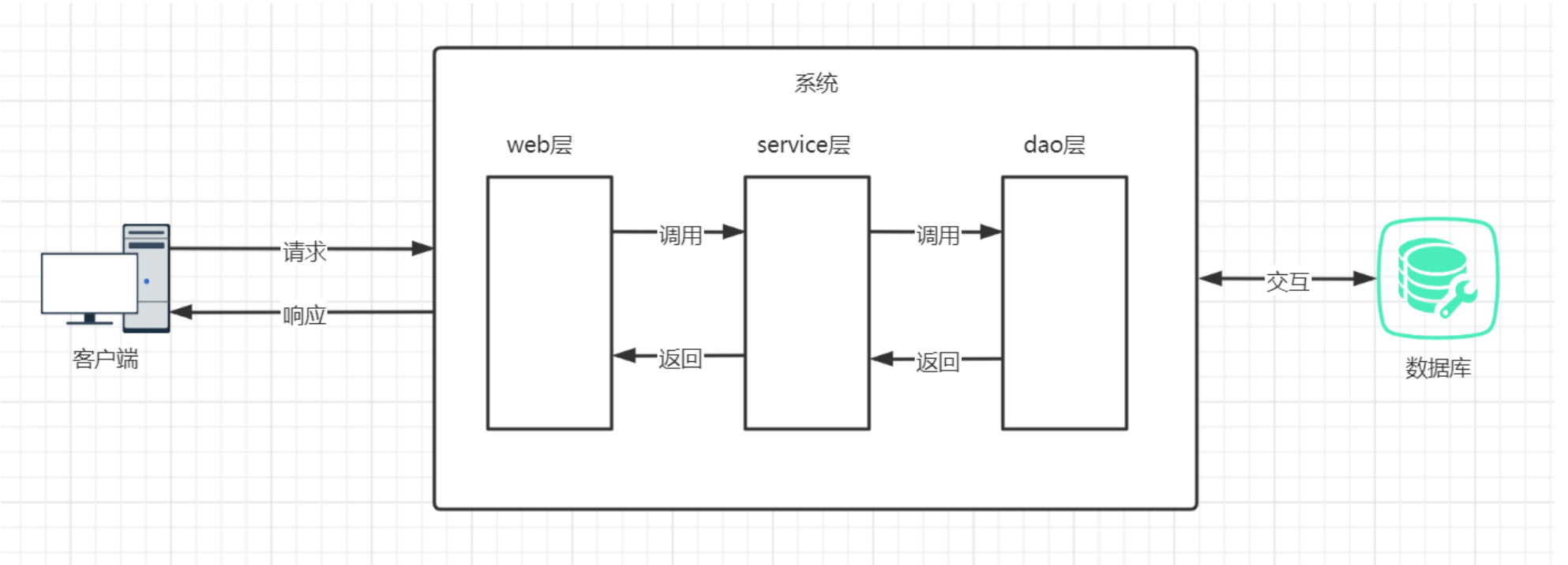
- 采用螺旋模型需要具有相当丰富的风险评估经验和专门知识
- 在风险较大的项目开发中，如果未能够及时标识风险，势必造成重大损失
- 过多的迭代次数会增加开发成本，延迟提交时间

架构

在当前项目中，我们采用之前介绍过的三层架构：

- web层，负责和客户端交互
- service层，负责处理业务逻辑功能
- dao层，负责和数据库交互

如图：



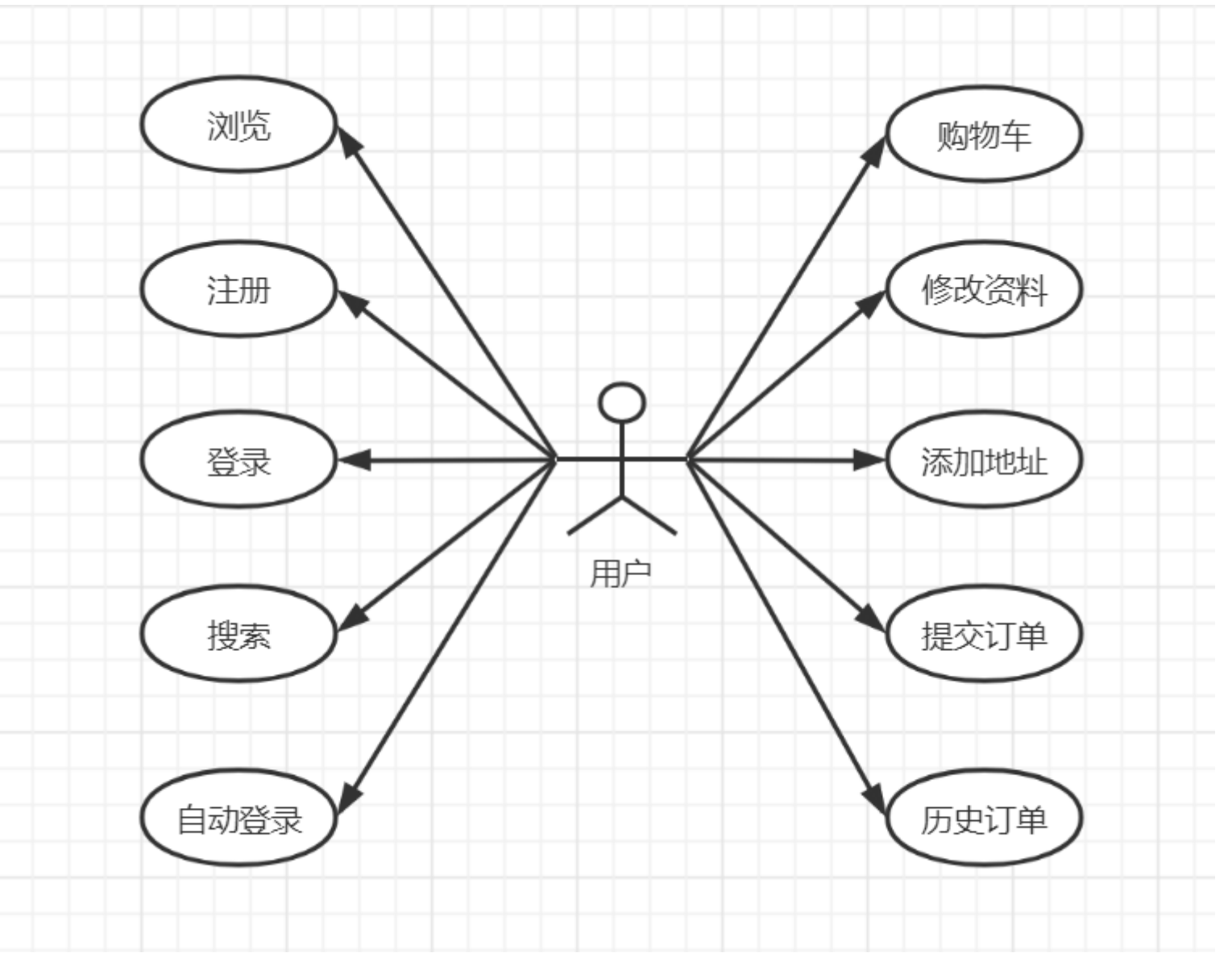
技术

该项目中所采用的技术：

- 前端技术
    - html、css、js、ajax、jquery
- 后端技术
    - servlet、jsp、jdbc、mybaits
- 数据库
    - oracle、mysql

用例

系统中，用户核心功能的用例图：



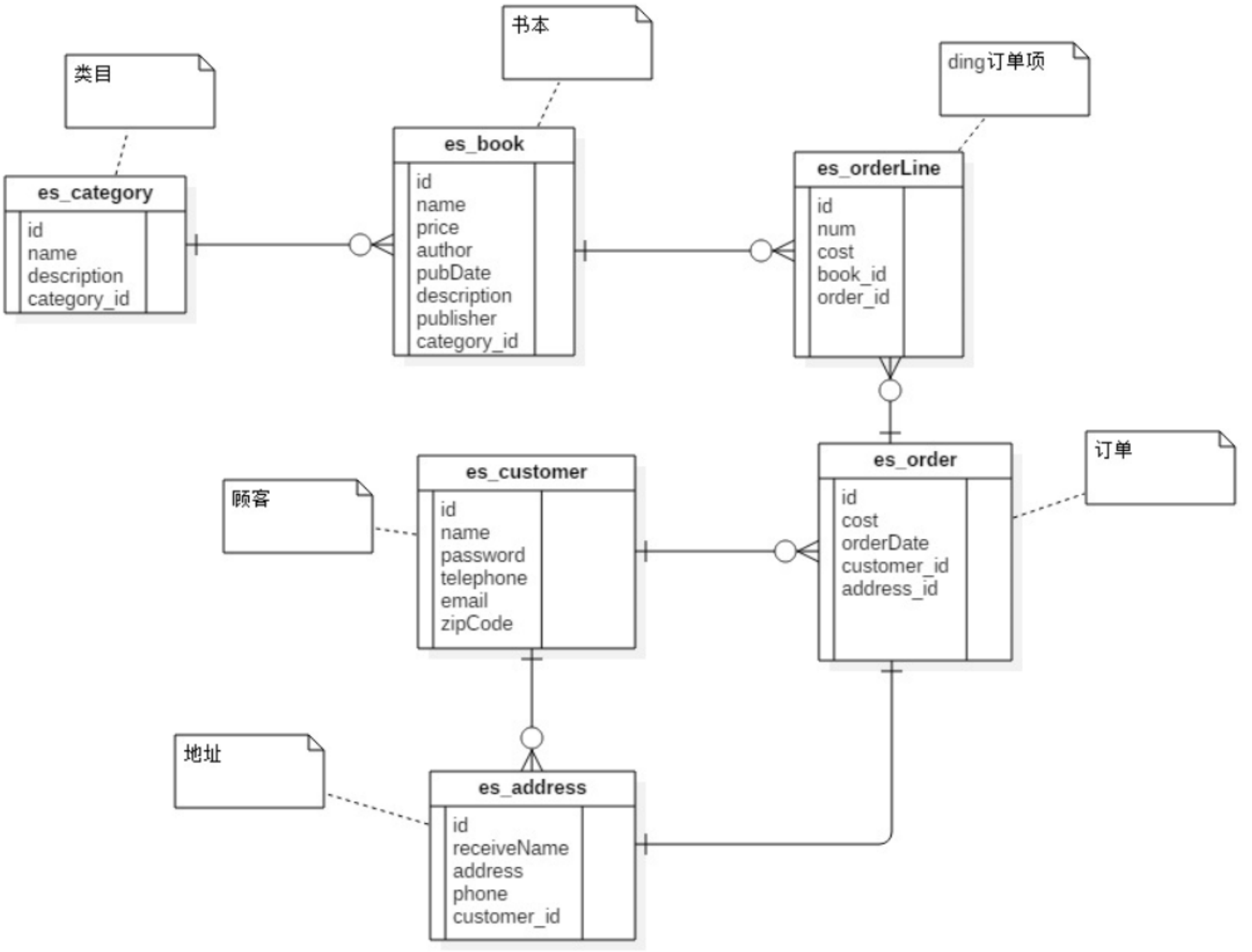
功能

系统中一些主要功能说明：

- 注册
    - 需要判断注册的用户名是否已经在数据库中存在
    - 最后将注册信息保存到数据库中
- 登录
    - 查看登陆的用户名是否存在
    - 查看密码是否正确
- 购物车
    - 增加订单项
    - 查看购物车
    - 删除订单项
    - 修改订单项
- 订单
    - 查看订单
    - 提交订单
    - 删除订单

ER图

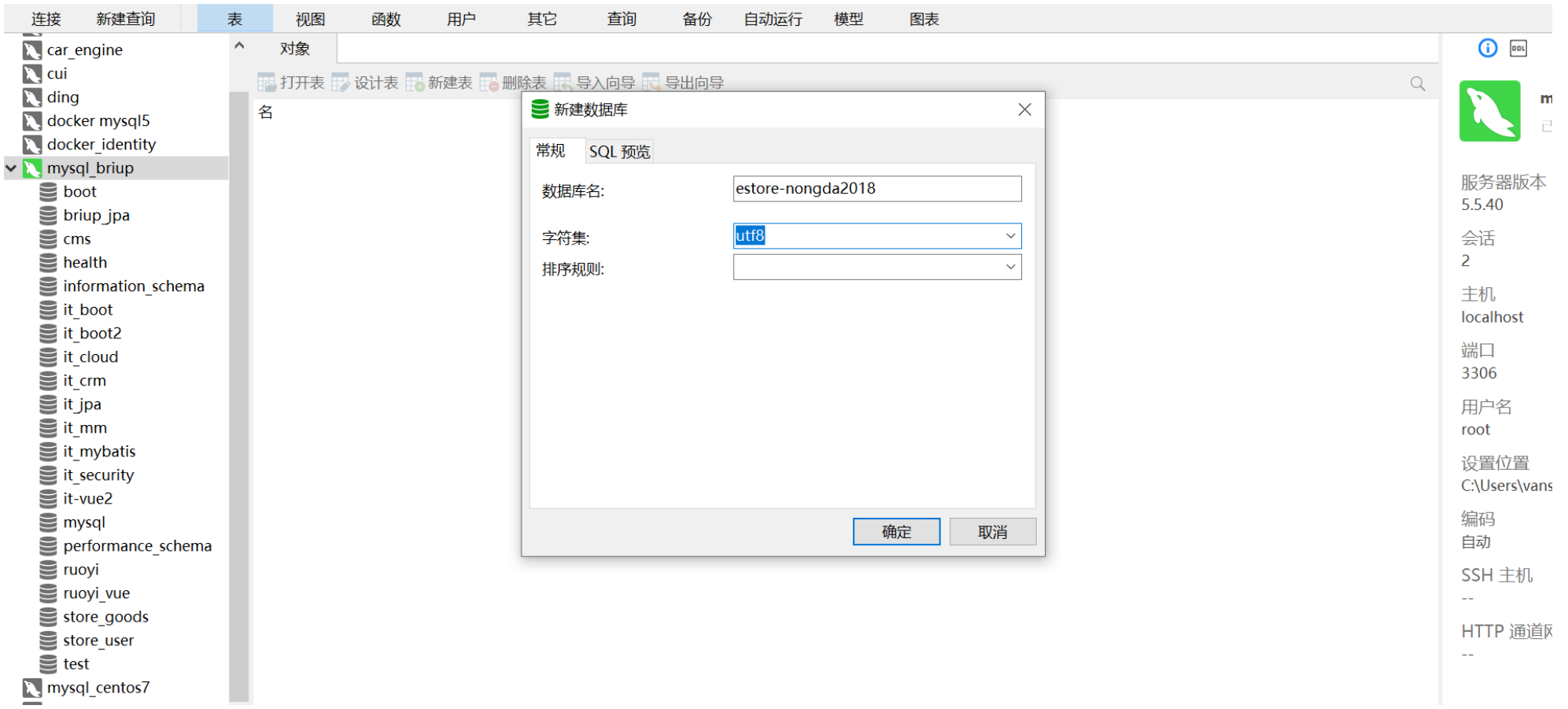
系统中的ER图：


书本
类目
ding订单项
es_book
id
name
price
author
pubDate
description
publisher
category_id
es_category
id
name
description
category_id
es_orderLine
id
num
cost
book_id
order_id
订单
es_order
id
cost
orderDate
customer_id
address_id
顾客
es_customer
id
name
password
telephone
email
zipCode
地址
es_address
id
receiveName
address
phone
customer_id

项目准备

数据库

导入表



数据字典

该系统的数据字典如下：

es_customer，用户表

列名	类型	约束	说明
id	int	primary key	编号
name	varchar	not null	用户名
password	varchar	not null	密码
zipCode	varchar	-	邮编
address	varchar	-	地址
telephone	varchar	-	电话
email	varchar	-	邮箱

es_book，书籍表

列名	类型	约束	说明
id	int	primary key	书籍编号
name	varchar	not null	书籍名字
price	varchar	not null	书籍价格
author	varchar	-	书籍作者
publisher	varchar	-	书籍出版社
pubDate	date	-	发布时间
description	varchar	-	书籍描述

列名	类型	约束	说明
image	varchar	-	书籍图片
category_id	int	foreign key	书籍类别编号

es_category，书籍分类表

列名	类型	约束	说明
id	int	primary key	书籍分类编号
name	varchar	not null	书籍分类名字
description	varchar	not null	书籍分类描述
parent_id	int	foreign key	父类别编号

es_orderform，订单表

列名	类型	约束	说明
id	int	primary key	订单编号
cost	double	not null	订单金额
orderDate	date	not null	订单时间
shopAddress_id	int	foreign key	收货地址编号
customer_id	int	foreign key	客户编号

es_orderline，订单项表

列名	类型	约束	说明
id	int	primary key	订单项编号
num	int	not null	商品数量
cost	date	not null	订单项金额
book_id	int	foreign key	书籍编号
orderForm_id	int	foreign key	订单编号

es_shopaddress，收货地址表

列名	类型	约束	说明
id	int	primary key	收货地址编号
receiveName	varchar	not null	收货人姓名
address	varchar	not null	收货人地址
phone	varchar	not null	收货人电话
customer_id	int	foreign key	客户编号

表分析

所有表都没有物理外键，而是逻辑外键

es_customer

- 顾客表 一个顾客有一个地址

es_shopaddress

- 地址表 顾客外键

es_category

- 商品分类表 一个分类有多个商品
- 一级分类/二级分类(自关联)
  - 二级分类的parentId是一级分类的id
  - 一级分类: `select * from category where parent_id = 0`
  - 二级分类: `select * from category where parent_id = 一级分类id*`

es_book

- 商品表 分类外键

es_orderline

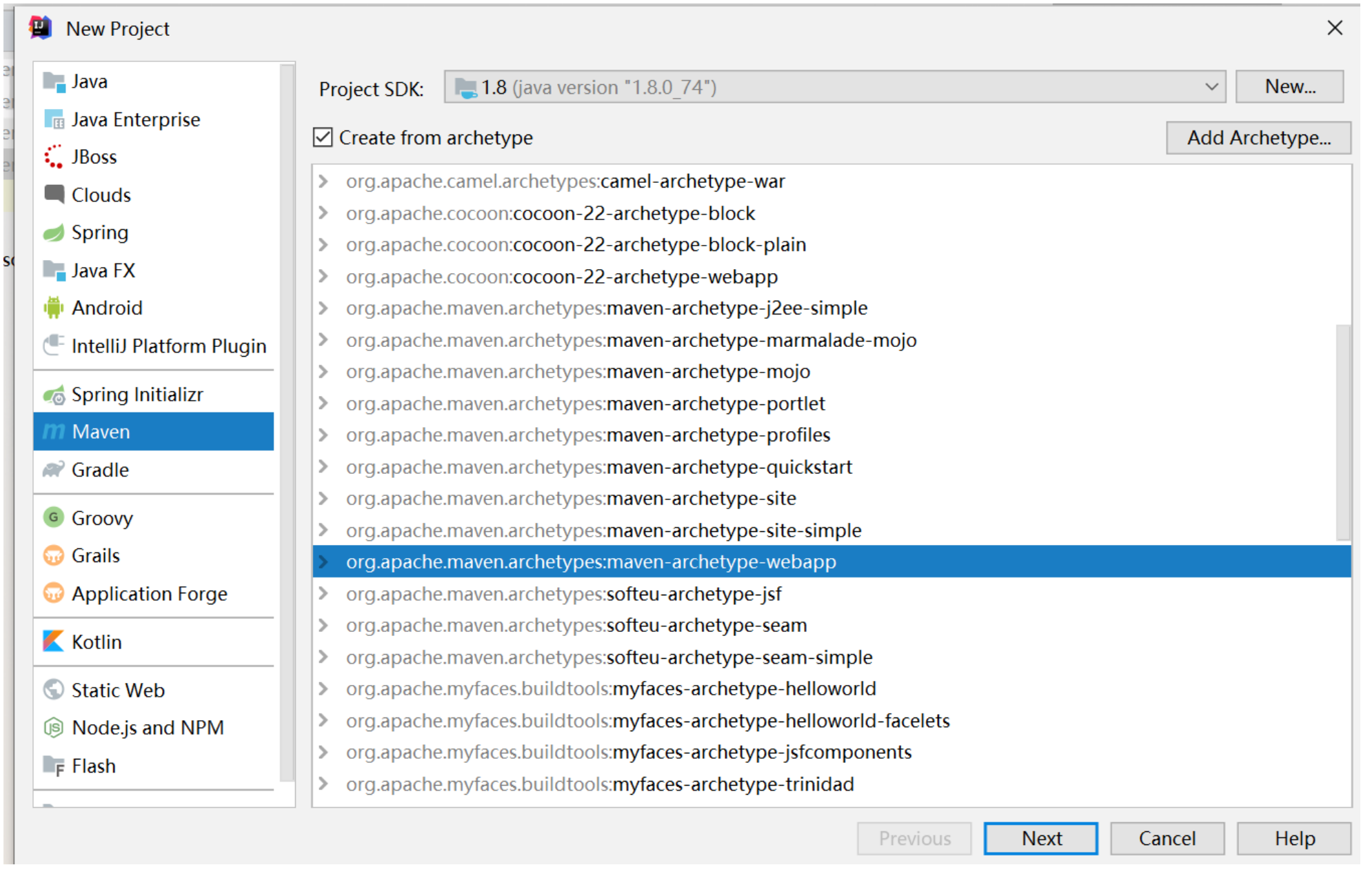
- 订单项 购买一次就是一个订单项

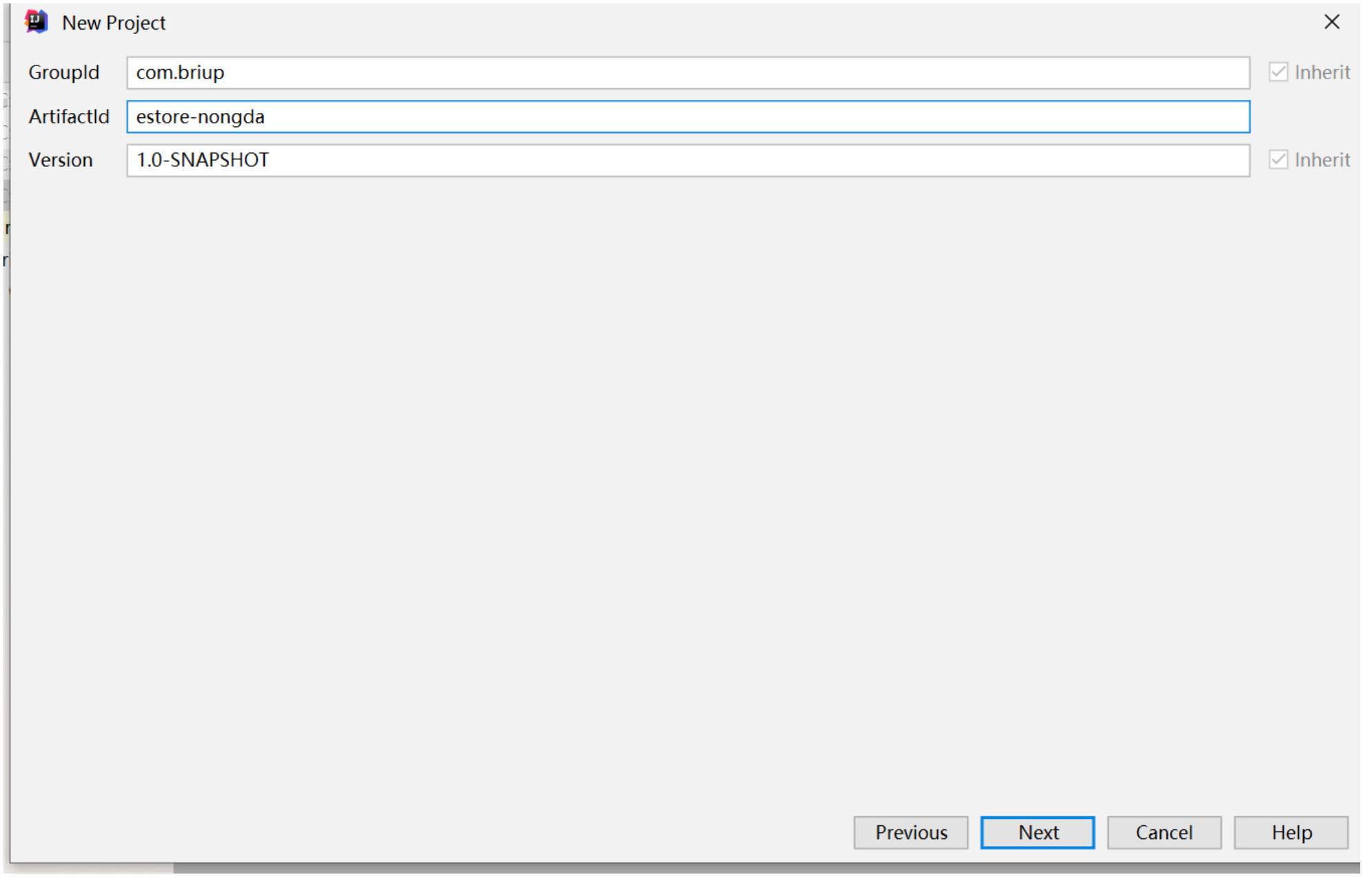
es_order

- 订单 一个订单有多个订单项

创建项目

1.选择骨架


New Project
Java
Java Enterprise
JBoss
Clouds
Spring
Java FX
Android
IntelliJ Platform Plugin
Spring Initializr
Maven
Gradle
Groovy
Grails
Application Forge
Kotlin
Static Web
Node.js and NPM
Flash
Project SDK:
1.8 (java version "1.8.0_74")
New...
Create from archetype
Add Archetype...
org.apache.camel.archetypes:camel-archetype-war
org.apache.cocoon:cocoon-22-archetype-block
org.apache.cocoon:cocoon-22-archetype-block-plain
org.apache.cocoon:cocoon-22-archetype-webapp
org.apache.maven.archetypes:maven-archetype-j2ee-simple
org.apache.maven.archetypes:maven-archetype-marmalade-mojo
org.apache.maven.archetypes:maven-archetype-mojo
org.apache.maven.archetypes:maven-archetype-portlet
org.apache.maven.archetypes:maven-archetype-profiles
org.apache.maven.archetypes:maven-archetype-quickstart
org.apache.maven.archetypes:maven-archetype-site
org.apache.maven.archetypes:maven-archetype-site-simple
org.apache.maven.archetypes:maven-archetype-webapp
org.apache.maven.archetypes:softeu-archetype-jsf
org.apache.maven.archetypes:softeu-archetype-seam
org.apache.maven.archetypes:softeu-archetype-seam-simple
org.apache.myfaces.buildtools:myfaces-archetype-helloworld
org.apache.myfaces.buildtools:myfaces-archetype-helloworld-facelets
org.apache.myfaces.buildtools:myfaces-archetype-jsfcomponents
org.apache.myfaces.buildtools:myfaces-archetype-trinidad
Previous
Next
Cancel
Help


New Project
GroupId
com.briup
Inherit
ArtifactId
estore-nongda
Version
1.0-SNAPSHOT
Inherit
Previous
Next
Cancel
Help

New Project

Maven home directory: E:/soft-it/apache-maven-3.6.3 ...

(Version: 3.6.3)

User settings file: E:\soft-it\apache-maven-3.6.3\conf\settings.xml Override

Local repository: E:\soft-it\maven_repsoitory Override

Properties

groupId	com.briup
artifactId	estore-nongda
version	1.0-SNAPSHOT
archetypeGroupId	org.apache.maven.archetypes
archetypeArtifactId	maven-archetype-webapp
archetypeVersion	RELEASE

Previous Next Cancel Help

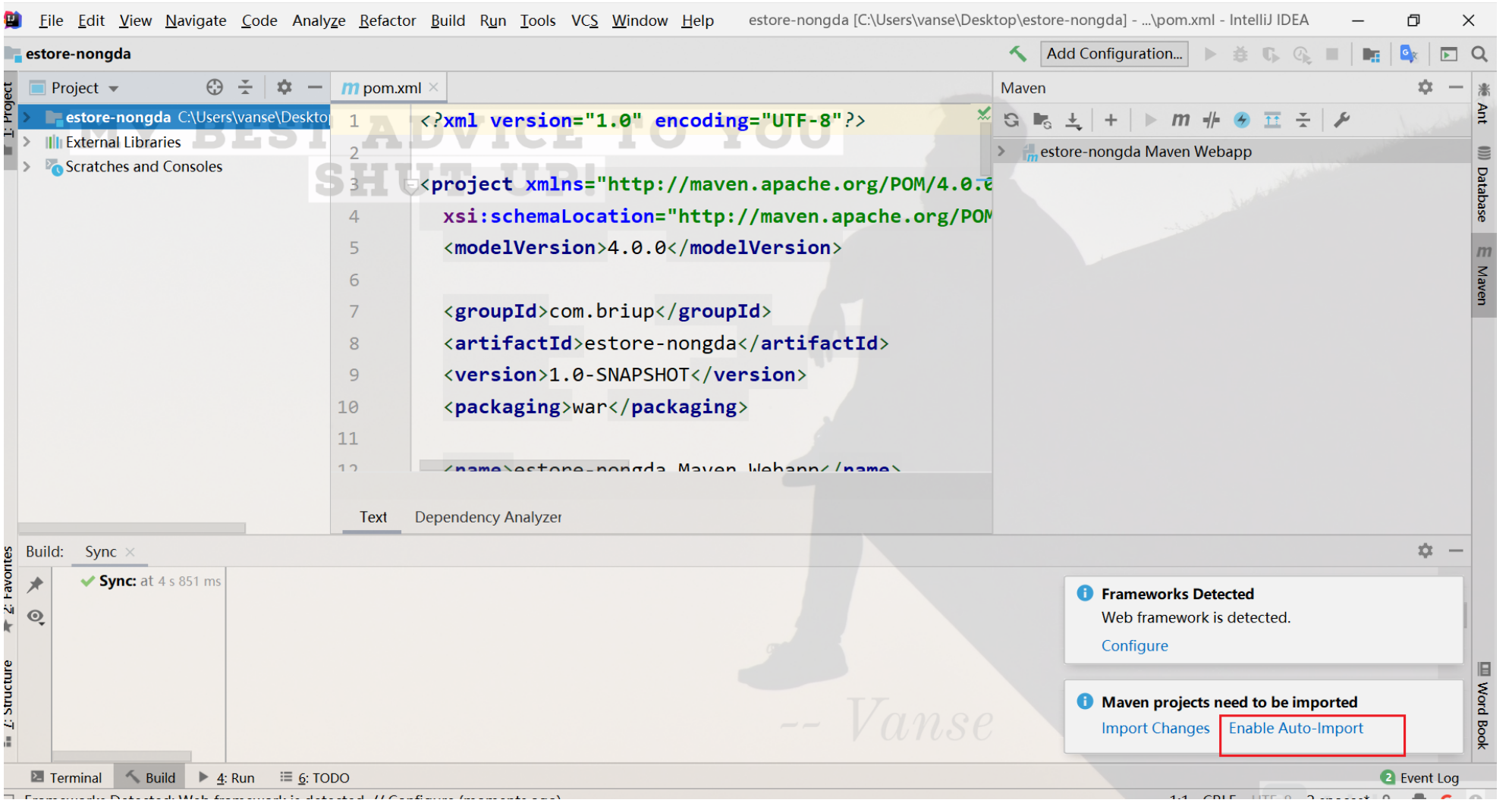
New Project

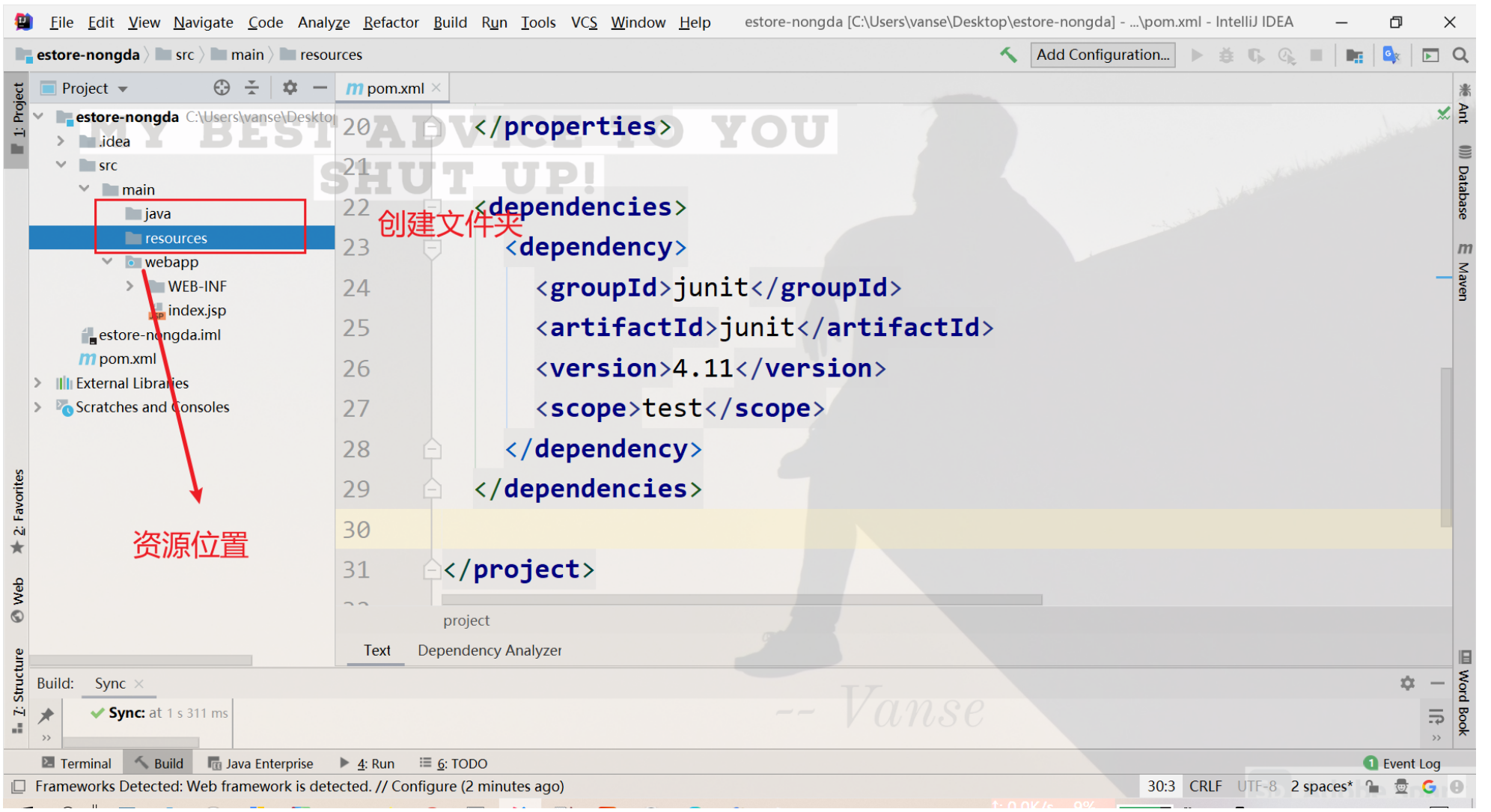
Project name: estore-nongda

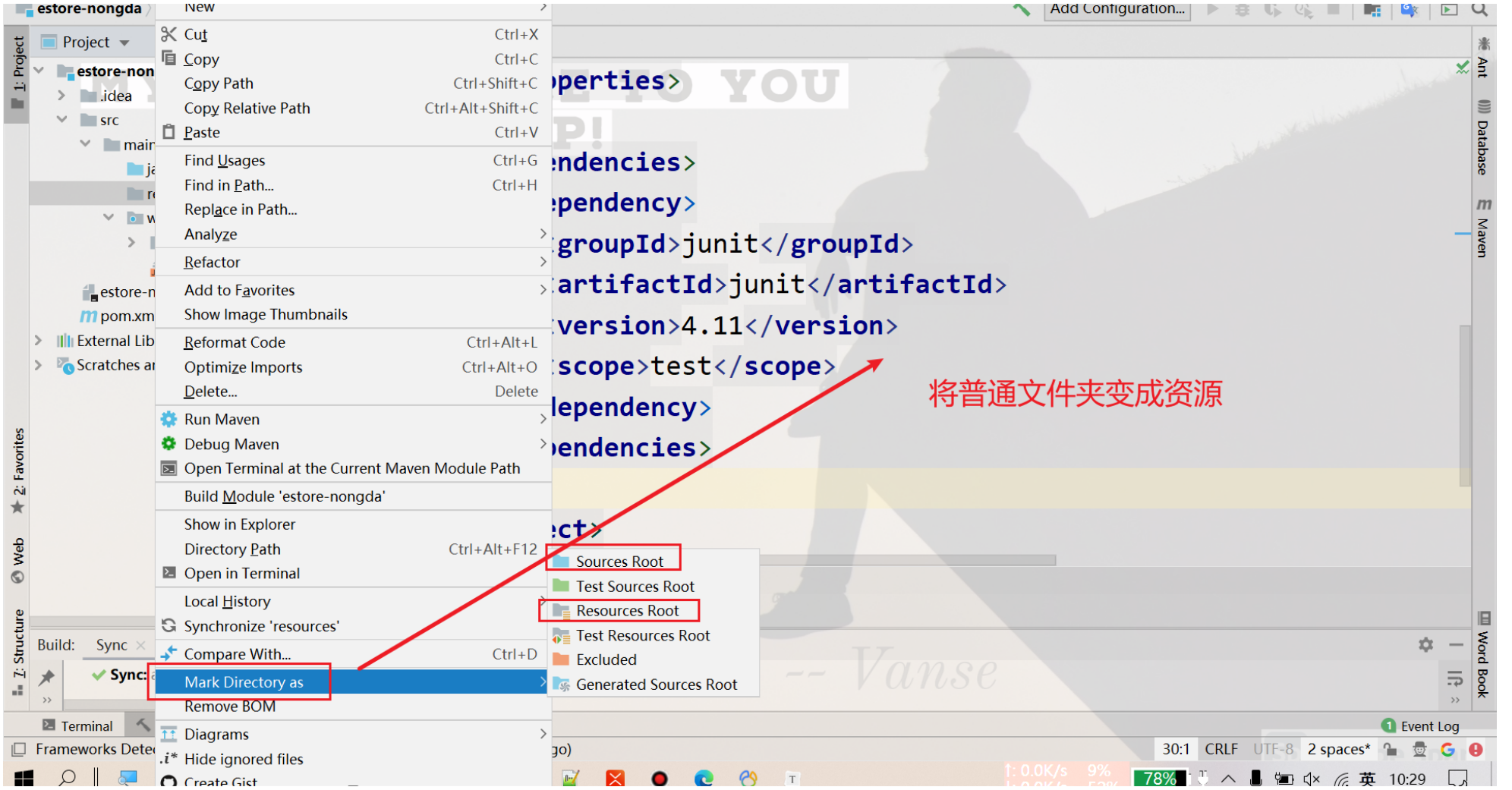
Project location: C:\Users\vanse\Desktop\estore-nongda ...

More Settings

Previous Finish Cancel Help


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
estore-nongda [C:\Users\vanse\Desktop\estore-nongda] - ...\pom.xml - IntelliJ IDEA
estore-nongda
Add Configuration...
Project
estore-nongda C:\Users\vanse\Deskto
External Libraries
Scratches and Consoles
pom.xml
1 <?xml version="1.0" encoding="UTF-8"?>
3 <project xmlns="http://maven.apache.org/POM/4.0.0
4 xsi:schemaLocation="http://maven.apache.org/POM
5 <modelVersion>4.0.0</modelVersion>
7 <groupId>com.briup</groupId>
8 <artifactId>estore-nongda</artifactId>
9 <version>1.0-SNAPSHOT</version>
10 <packaging>war</packaging>
Text
Dependency Analyzer
Maven
estore-nongda Maven Webapp
Build:
Sync
Sync: at 4 s 851 ms
Frameworks Detected
Web framework is detected.
Configure
Maven projects need to be imported
Import Changes
Enable Auto-Import
Terminal
Build
4: Run
6: TODO
Event Log
MY BEST ADVICE TO YOU
SHUT UP!
-- Vanse


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
estore-nongda [C:\Users\vanse\Desktop\estore-nongda] - ...\pom.xml - IntelliJ IDEA
estore-nongda src main resources
Add Configuration...
Project
estore-nongda C:\Users\vanse\Deskto
.idea
src
main
java
resources
webapp
WEB-INF
index.jsp
estore-nongda.iml
pom.xml
External Libraries
Scratches and Consoles
创建文件夹
资源位置
pom.xml
20 </properties>
22 <dependencies>
23 <dependency>
24 <groupId>junit</groupId>
25 <artifactId>junit</artifactId>
26 <version>4.11</version>
27 <scope>test</scope>
28 </dependency>
29 </dependencies>
30
31 </project>
project
Text
Dependency Analyzer
Build:
Sync
Sync: at 1 s 311 ms
Terminal
Build
Java Enterprise
4: Run
6: TODO
Event Log
Frameworks Detected: Web framework is detected. // Configure (2 minutes ago)
30:3
CRLF
UTF-8
2 spaces*
MY BEST ADVICE TO YOU
SHUT UP!
-- Vanse



- java -> Sources Root
- resources-> Resources Root

2.pom.xml

> **注意: 如果是mysql8 最好使用mysql8的驱动**
>
> **同时: 驱动为 com.mysql.cj.jdbc.Driver**

```
<properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- 引入servlet3.1的依赖 -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>

        <!-- 引入jstl的依赖，jsp页面中会使用到 -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
```

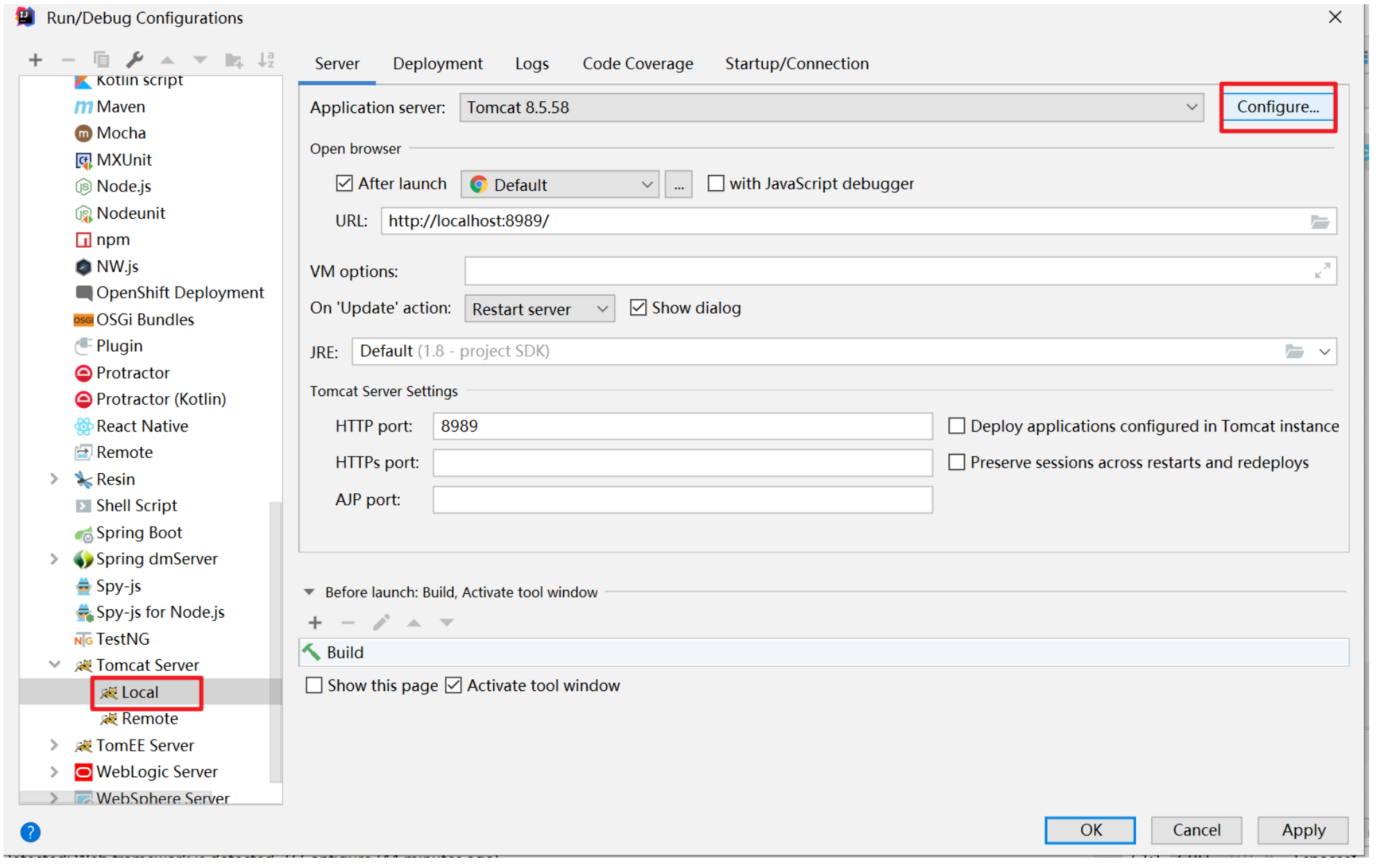
```
        <version>1.2</version>
    </dependency>

    <!-- 引入Mybatis的依赖 -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.4.6</version>
    </dependency>

    <dependency>
        <groupId>com.oracle.ojdbc</groupId>
        <artifactId>ojdbc8</artifactId>
        <version>19.3.0.0</version>
    </dependency>

    <!--
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.38</version>
    </dependency>
     -->
    <!-- 日志框架log4j的依赖 -->
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.17</version>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>


    <dependency>
        <groupId>com.alipay.sdk</groupId>
        <artifactId>alipay-sdk-java</artifactId>
        <version>4.8.10.ALL</version>
    </dependency>

    <!--json-->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.11.3</version>
    </dependency>

    <dependency>
```

```
            <groupId>cn.hutool</groupId>
            <artifactId>hutool-all</artifactId>
            <version>5.8.3</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>4.1.2</version>
        </dependency>
    </dependencies>
```

完善结构

配置tomcat

添加tomcat



部署项目到tomcat

Run/Debug Configurations

Server | Deployment | Logs | Code Coverage | Startup/Connection

Deploy at the server startup

Nothing to deploy

Artifact...
External Source...

Before launch: Build, Activate tool window

Build

Show this page Activate tool window

OK | Cancel | Apply

Select Artifacts to Deploy

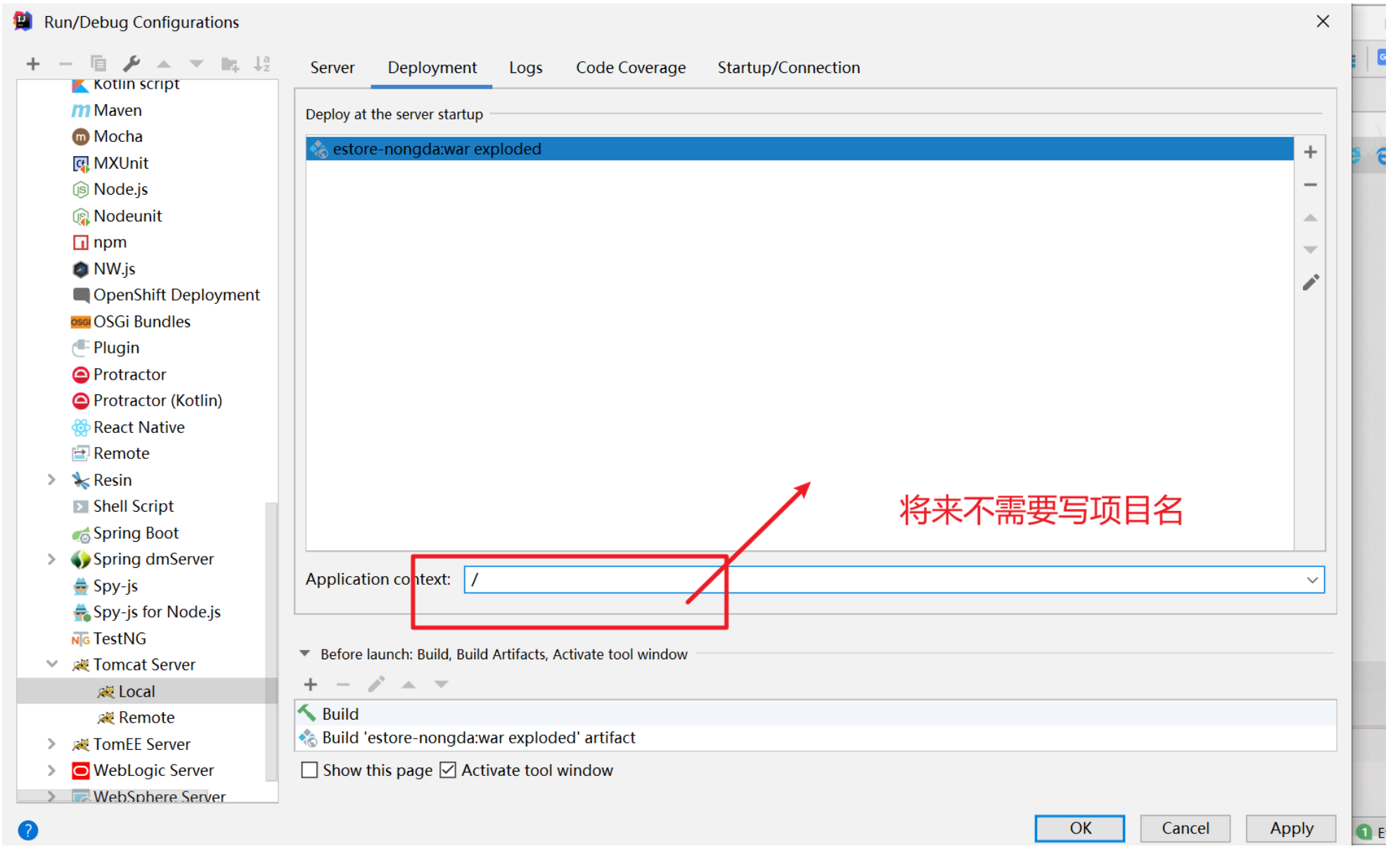
Selected artifacts will be deployed at server startup

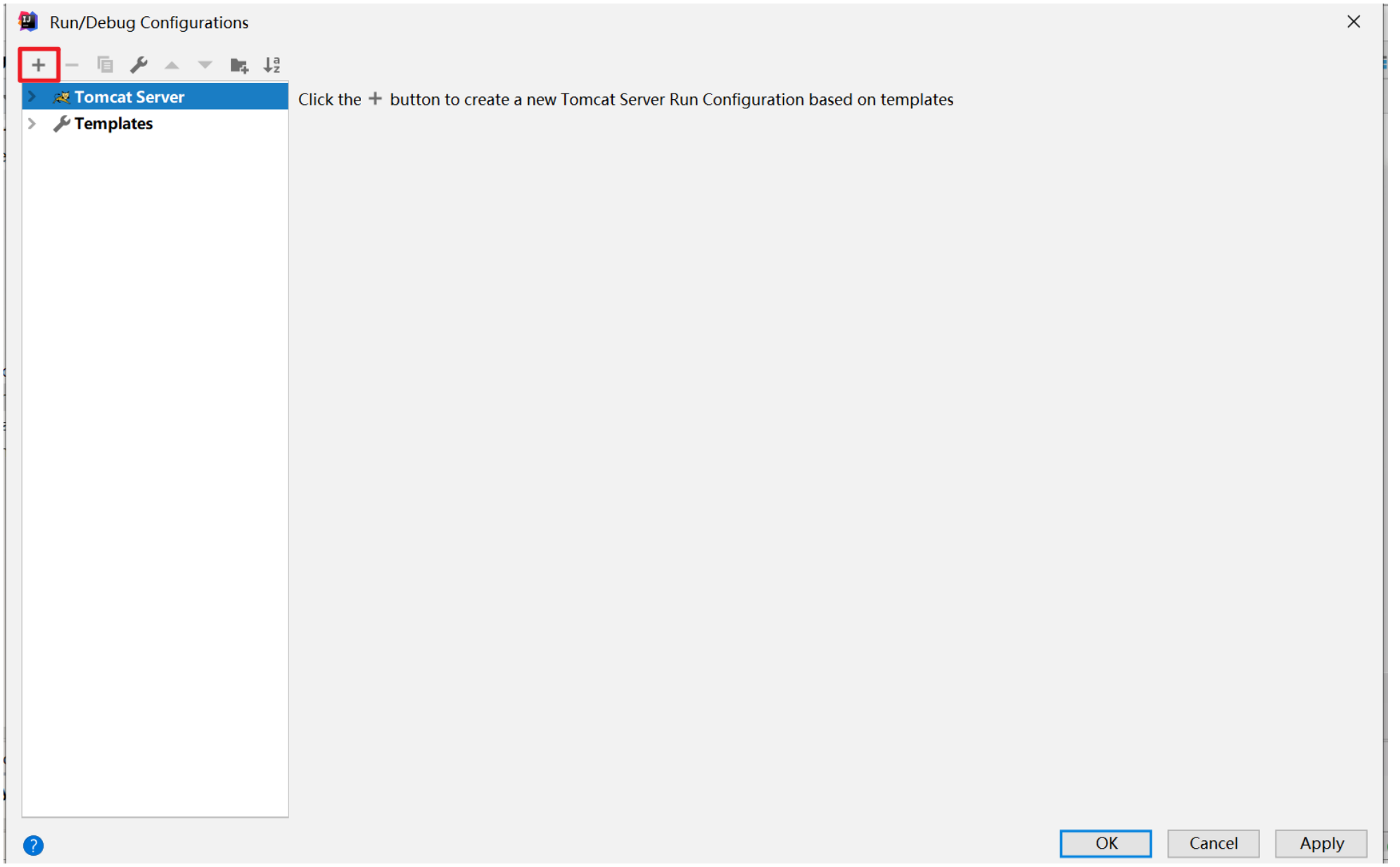
estore-nongda:war

estore-nongda:war exploded

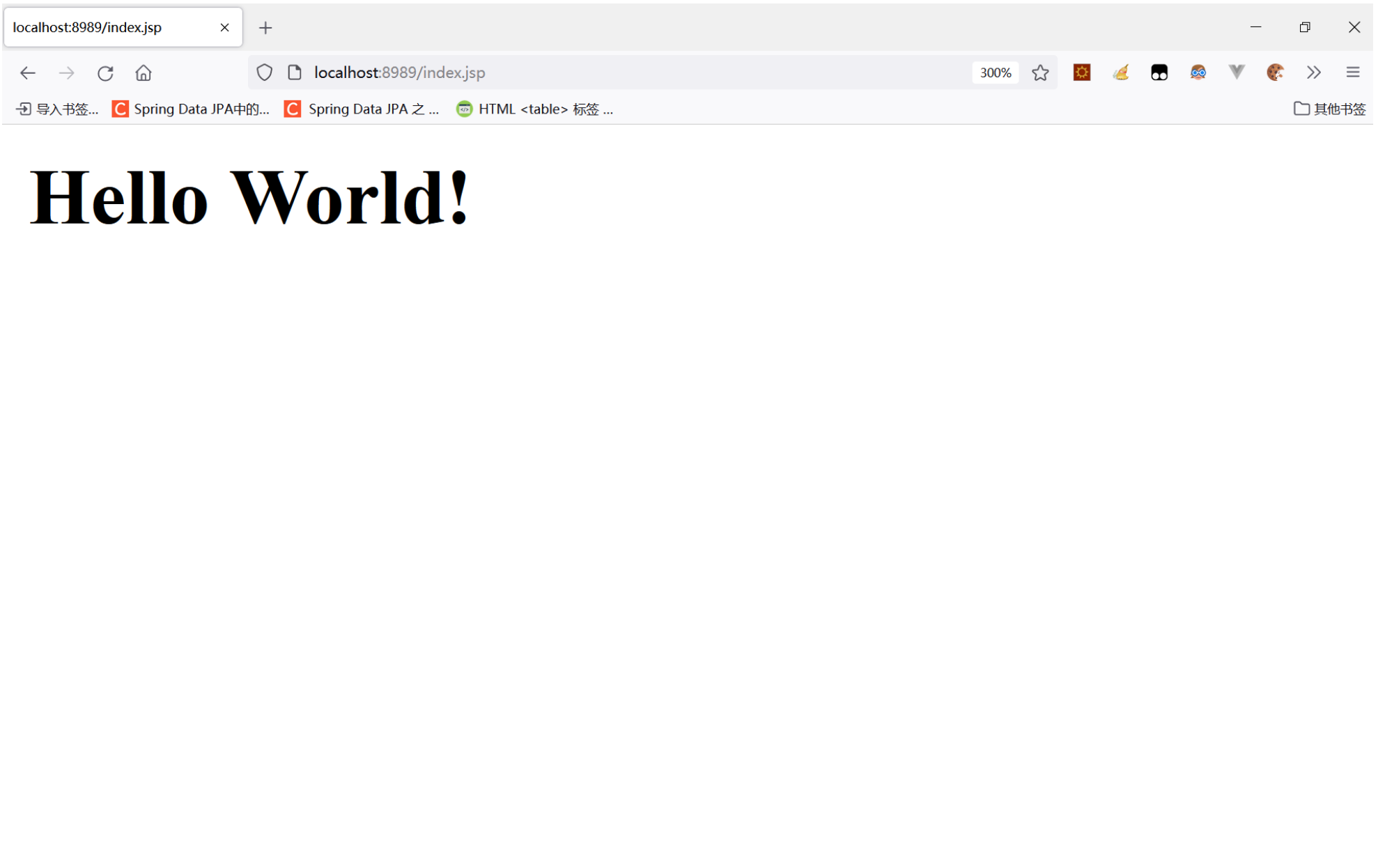
OK | Cancel

配置项目部署路径


Run/Debug Configurations
Server
Deployment
Logs
Code Coverage
Startup/Connection
Deploy at the server startup
estore-nongda:war exploded
将来不需要写项目名
Application context:
/
Before launch: Build, Build Artifacts, Activate tool window
Build
Build 'estore-nongda:war exploded' artifact
Show this page
Activate tool window
OK
Cancel
Apply
Kotlin script
Maven
Mocha
MXUnit
Node.js
Nodeunit
npm
NW.js
OpenShift Deployment
OSGi Bundles
Plugin
Protractor
Protractor (Kotlin)
React Native
Remote
Resin
Shell Script
Spring Boot
Spring dmServer
Spy-js
Spy-js for Node.js
TestNG
Tomcat Server
Local
Remote
TomEE Server
WebLogic Server
WebSphere Server


Run/Debug Configurations
Tomcat Server
Templates
Click the + button to create a new Tomcat Server Run Configuration based on templates
OK
Cancel
Apply

启动tomcat


localhost:8989/index.jsp
localhost:8989/index.jsp
300%
导入书签...
Spring Data JPA中的...
Spring Data JPA 之 ...
HTML <table> 标签 ...
其他书签
Hello World!

Run/Debug Configurations

Name: Tomcat 8.5.58

Share through VCS

Tomcat Server
Tomcat 8.5.58
Templates

Server | Deployment | Logs | Code Coverage | Startup/Connection

Application server: Tomcat 8.5.58 Configure...

Open browser

After launch Firefox ... with JavaScript debugger

URL: http://localhost:8989/

VM options:

On 'Update' action: Update resources Show dialog

On frame deactivation: Update resources

JRE: Default (1.8 - project SDK)

Tomcat Server Settings

HTTP port: 8989

HTTPs port:

JMX port: 1099

AJP port:

Deploy applications configured in Tomcat instance

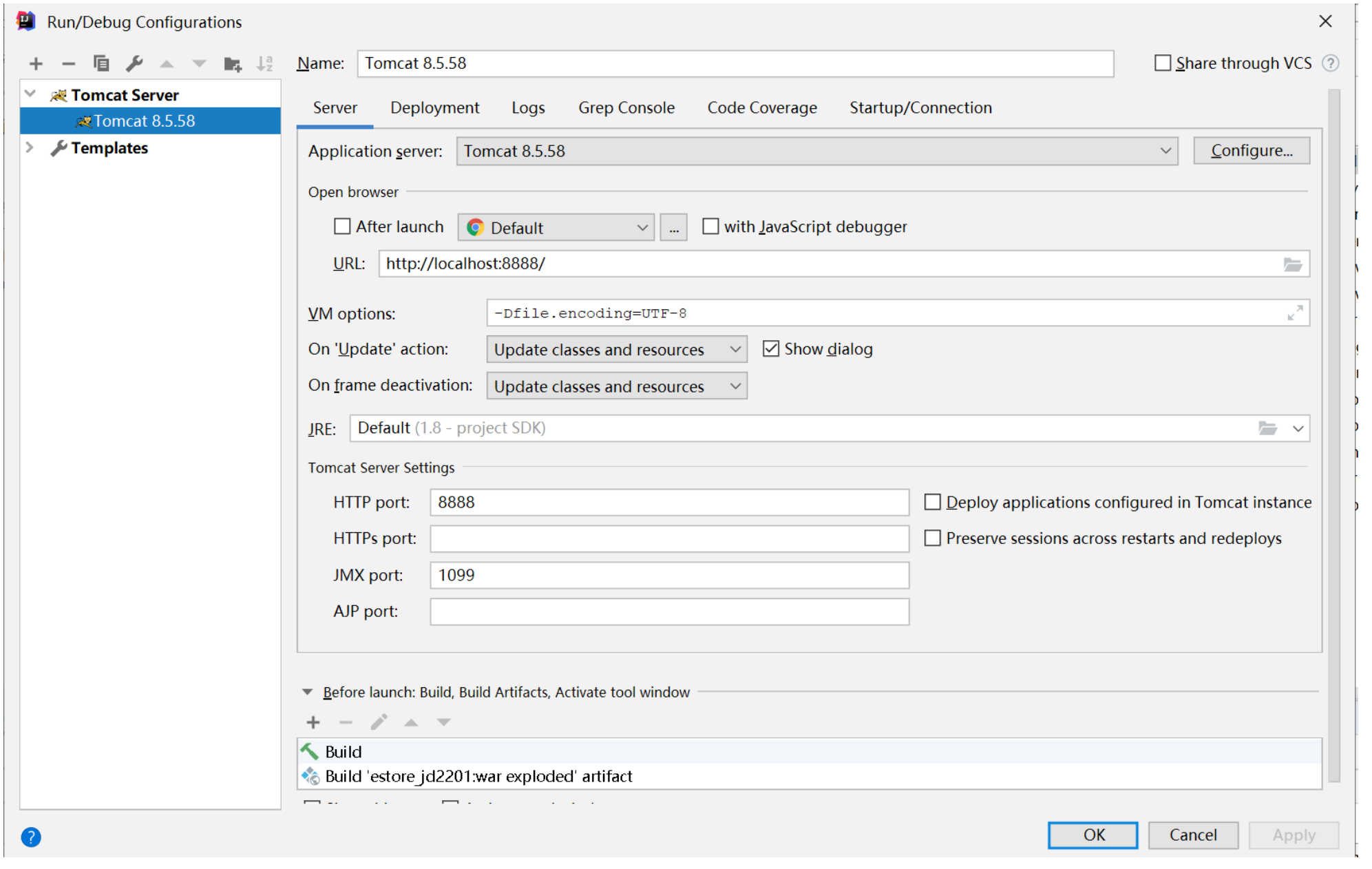
Preserve sessions across restarts and redeploys

Before launch: Build, Build Artifacts, Activate tool window

Build

Build 'estore-nongda:war exploded' artifact

OK Cancel Apply



测试连接

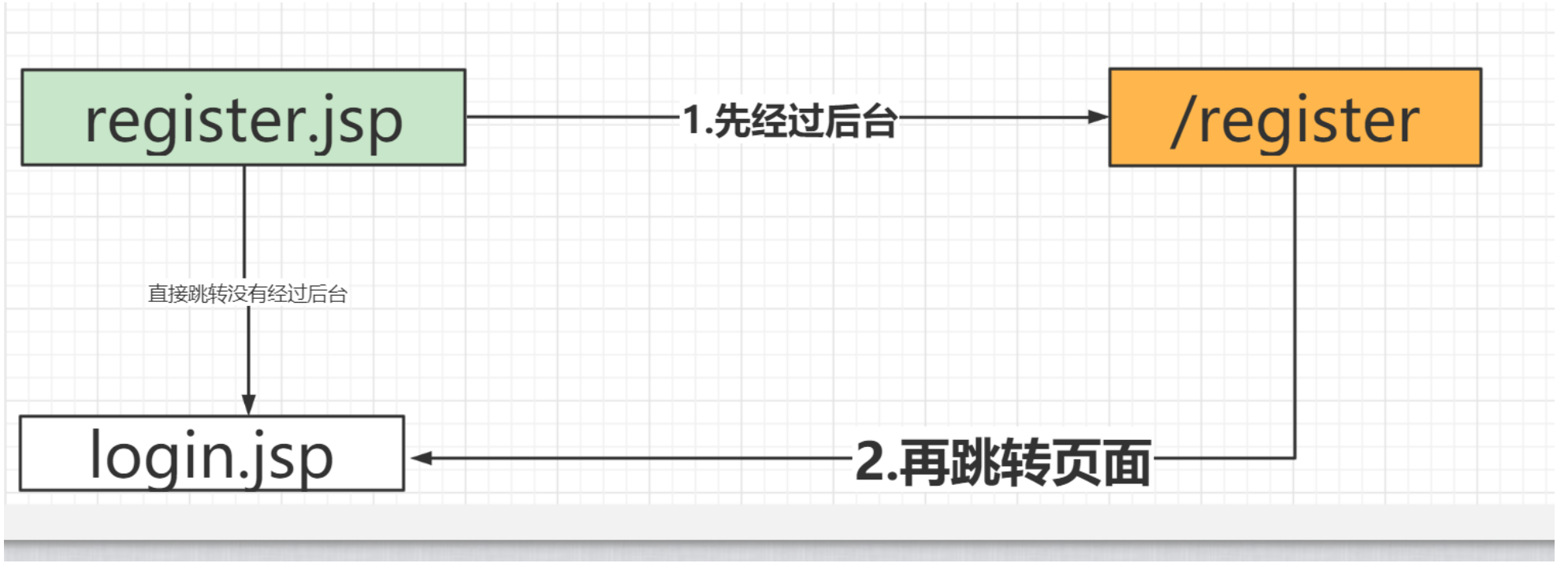
参考案例

顾客模块

注册

分析

- 获取页面参数
  - 判断name和password是否为空
  - 判断name是否存在
- 注册用户
  - 正常: 跳转登录页面
  - 错误: 获取错误信息,重新跳转注册页面



RegisterServlet

注意: 不同的请求方式可以访问同一接口有不同功能

```
package com.briup.estore.web.servlet;

import com.briup.estore.entity.Customer;
import com.briup.estore.service.IUserService;
import com.briup.estore.service.impl.UserServiceImpl;
import org.apache.taglibs.standard.tag.common.sql.DataSourceUtil;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/5/31-05-31-10:33
 * @Description:  用户注册接口 (接收/响应数据)
 */
@WebServlet("/userRegister")
public class UserRegisterServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 1.接收参数 并封装Customer对象
        String name = req.getParameter("name");
        String password = req.getParameter("password");
        String zipCode = req.getParameter("zipCode");
```

```
        String address = req.getParameter("address");
        String telephone = req.getParameter("telephone");
        String email = req.getParameter("email");
        Customer customer = new
Customer(name,password,zipCode,address,telephone,email);
        // 2.交给service判断 用户名不能为空
        IUserService userService = new UserServiceImpl();
        // 3.判断是否注册成功
        String path="login.jsp";
        try {
            userService.register(customer);
        } catch (Exception e) {
            e.printStackTrace(); // 打印异常信息
            // 失败 重新注册  请求转发
            // 将该错误信息存到request中,将来页面取出
            req.getSession().setAttribute("error","注册失败 " +e.getMessage());
            path = "register.jsp";
        }
    }
}
```

CustomerServiceImpl

```
package com.briup.estore.service.impl;

import com.briup.estore.dao.ICustomerDao;
import com.briup.estore.entity.Customer;
import com.briup.estore.service.IUserService;
import com.briup.estore.utils.MD5Util;
import com.briup.estore.utils.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/5/31-05-31-11:01
 * @Description：顾客业务实现
 */
public class UserServiceImpl implements IUserService {
    SqlSession session = MyBatisSqlSessionFactory.openSession();
    ICustomerDao customerDao = session.getMapper(ICustomerDao.class);

     @Override
    public void register(Customer customer) {
        String name = customer.getName();
        // 1.校验用户
            // 1.用户名不能为空
        if (name == null || "".equals(name.trim())) {
```

```
            // 参数是message new RuntimeException(message);
            throw new RuntimeException("用户名不能为空");
        }
            // 2.用户名不能已经在数据库存在 (携带用户名去数据库查)
        Customer cusomerFromDB = findByName(name);
        if (cusomerFromDB != null) {
            throw new RuntimeException("用户名已经存在");
        }
        // 2.注册用户(调用dao)
        customer.setPassword(customer.getPassword());
        customerDao.register(customer);
        // 注意事务的提交
        session.commit();
    }

    /**
     *
     * @param name 用户名称
     * @return 该用户对象
     */
    public Customer findByName(String name){
        // 去数据库查
        return customerDao.findByName(name);
    }
}
```

注意: 注册需要提交 commit

CustomerMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.briup.dao.CustomerMapper">

    <resultMap type="Customer" id="CustomerResult">
        <id property="id" column="id"/>
        <result property="name" column="name"/>
        <result property="password" column="password"/>
        <result property="zipCode" column="zipCode"/>
        <result property="address" column="address"/>
        <result property="telephone" column="telephone"/>
        <result property="email" column="email"/>
    </resultMap>

    <select id="findCustomerByName" parameterType="string"
resultMap="CustomerResult">
        select
```

```
            id,name,password,zipCode,address,telephone,email
        from
            es_customer
        where
            name = #{name}
    </select>

    <insert id="saveCustomer" parameterType="Customer" useGeneratedKeys="true"
keyProperty="id">

        insert into es_customer(name,password,zipCode,address,telephone,email)
        values(#{name},#{password},#{zipCode},#{address},#{telephone},#{email})

    </insert>

</mapper>
```

register.jsp

```
 <ul class="tabs">
     <c:choose>
         <c:when test="${empty error}">
             <li class="phone current"><a href="#">用户注册，请将信息填写完整</a></li>
         </c:when>
         <c:otherwise>
             <li class="phone current"><span style="color: red">${error}</span></li>
         </c:otherwise>
     </c:choose>
     <c:remove var="error" scope="session"/>
</ul>


<li>
    <span><b>*</b>用户名：</span>
    <input type="text" name="name" id="name"/>
    <span id="tip" style="width: 110px;text-align: justify;padding-left: 10px">
</span>
</li>
```

注意：1.idea骨架创建webapp 默认页面忽略了el表达式 2.引入jq必须写双标签

登录

分析

- 获取到账户密码
- 校验是否是账户错误还是密码错误
- 成功 跳转到商城首页
    - 失败 回退到登录页面 并提示错误信息

LoginServlet

```
package com.briup.estore.web.servlet;

import com.briup.estore.constant.EstoreConstant;
import com.briup.estore.entity.Customer;
import com.briup.estore.service.IUserService;
import com.briup.estore.service.impl.UserServiceImpl;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/5/31-05-31-15:42
 * @Description: 登录接口
 */
@WebServlet("/userLogin")
public class UserLoginServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 1.获取用户名和密码
        String name = req.getParameter("name");
        String password = req.getParameter("password");
        String auto = req.getParameter("auto");
        // 2.调用service校验用户名和密码
        IUserService userService = new UserServiceImpl();
        // 3.跳转页面
        String path = "index.jsp";
        HttpSession session = req.getSession();
        try {
            // 登录成功  跳转首页(重定向)
            Customer customer = userService.login(name, password);
            session.setAttribute("customer", customer);
        } catch (Exception e) {
            e.printStackTrace();
            // 登录失败  重新登录(请求转发 提示信息)
            path = "login.jsp";
            session.setAttribute("error", "登录失败 " +e.getMessage());
```

```
            }
            resp.sendRedirect(path);
        }
    }
```

CustomerServiceImpl

```
@Override
public Customer login(String name, String password) {
    // 校验用户名是否错误 这里没有校验密码
    Customer customer = customerMapper.findByName(name);
    if(customer == null){
        throw new RuntimeException("账户名有误");
    }
    // 校验错误是否错误
    if(!password.equals((customer.getPassword()))){
        throw new RuntimeException("密码有误");
    }

    return customer;
}
```

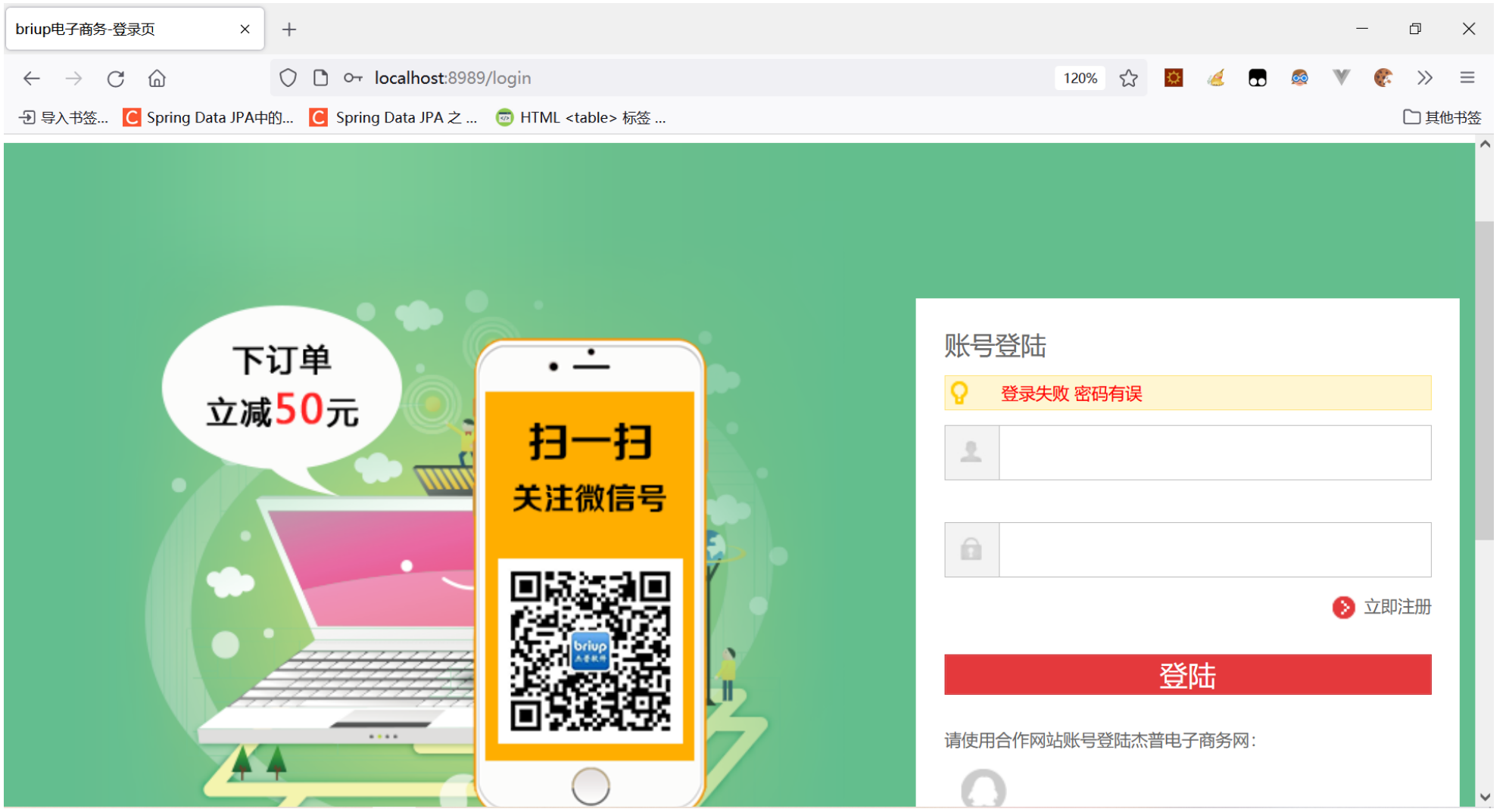
login.jsp

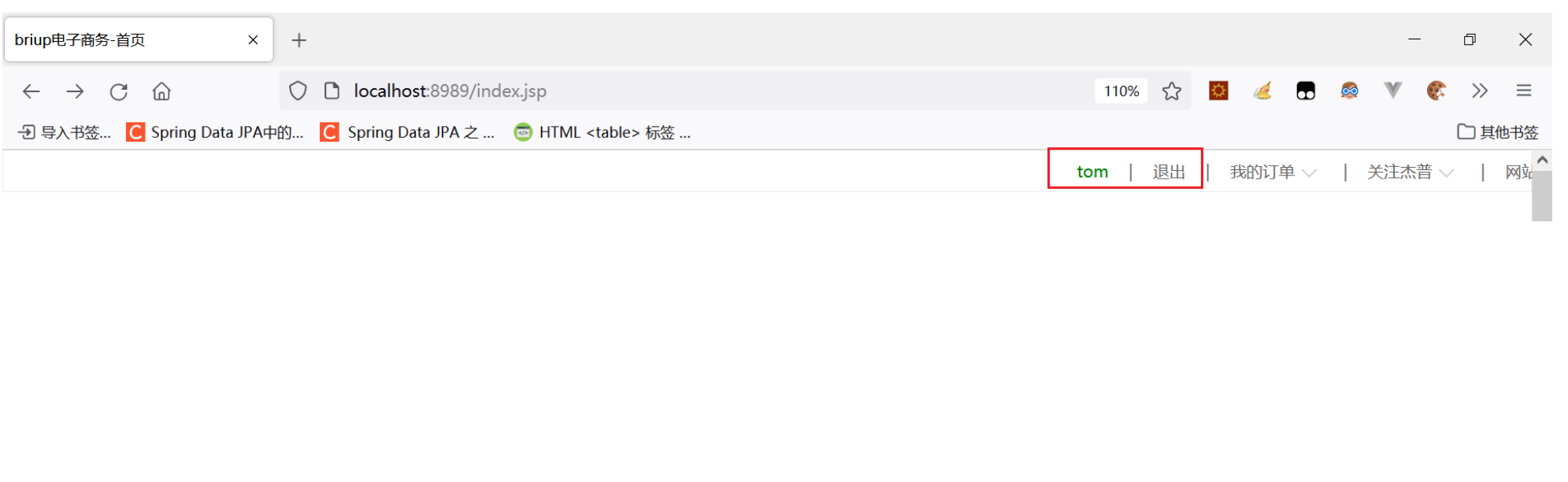
```
<form action="/userLogin" method="post">
    <h1>账号登陆</h1>
    <c:choose>
        <c:when test="${empty error}">
            <h2>公共场所请不要泄露您的密码，以防止账号丢失</h2>
        </c:when>
        <c:otherwise>
            <h2>${error}</h2>
        </c:otherwise>
    </c:choose>
    <c:remove var="error" scope="session"/>
```

c:remove

index.jsp

```
<%--登录后才显示退出--%>
<c:choose>
    <c:when test="${!empty customer}">
        <li><a href="#"><span style="color: green;font-weight:
bold">${customer.name}</span></a>|</li>
        <li><a href="/userLogout">退出</a>|</li>
    </c:when>
    <c:otherwise>
        <li><a href="login.jsp">请登录</a>|</li>
    </c:otherwise>
</c:choose>
<li><a href="orderList">我的订单<span class="jt_down"></span></a>|</li>
<li><a href="#">关注杰普<span class="jt_down"></span></a>|</li>
<li><a href="#">网站导航<span class="jt_down"></span></a></li>
```





注销

分析

- 如果注销用户,所有存在session的数据例如购物车都应该清除
- 销毁session即可

LogoutServlet

```
@WebServlet("/logoutServlet")
public class LogoutServlet extends HttpServlet{

    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {

        req.getSession().invalidate();

        String path = "/login.jsp";

        resp.sendRedirect(req.getContextPath()+path);
    }

}
```

ajax校验(作业)

为什么需要前后端都校验

- 前端显示友好
- 后端控制业务，防止页面操作问题

分析

- 输入框绑定离焦事件 发送ajax请求
- 后台根据用户名查找用户是否存在
- 响应提示信息

CheckServlet

```java
package com.briup.estore.web.servlet;

import com.briup.estore.dto.Result;
import com.briup.estore.entity.Customer;
import com.briup.estore.service.IUserService;
import com.briup.estore.service.impl.UserServiceImpl;
import com.fasterxml.jackson.databind.ObjectMapper;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/1-06-01-9:42
 * @Description: ajax校验用户名是否存在
 */
@WebServlet("/userCheck")
public class UserCheckServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        String name = req.getParameter("name");
        resp.setContentType("text/json;charset=utf-8");
        PrintWriter writer = resp.getWriter();

        IUserService userService = new UserServiceImpl();
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            Customer customer = userService.findByName(name);
            Result result = null;
            if (customer != null) {
                result = new Result("用户名已经存在","red");
            }else {
                result = new Result("用户名可用","green");
            }
            writer.write(objectMapper.writeValueAsString(result));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

register.jsp

注意:　改造之前的service findByName方法

```
<script type="text/javascript" src="js/jquery-1.12.3.min.js"></script>
<script type="text/javascript">
    // ajax校验
    $(function(){
        $("#name").on("blur",function () {
            var name = $(this).val();
            $.ajax({
                type: "GET",
                url: "/userCheck",
                data: "name="+name,
                dataType: "json",
                success: function (result) {

 $("#tip").text(result.msg).css({"color":result.color}).hide().slideDown(2000).slide
Up(2000);
                }
            })
        });
    })
</script>

 <li>
     <span><b>*</b>用户名：</span>
     <input type="text" name="name" id="name"/>
     <span id="tip" style="width: 110px;text-align: justify;padding-left:
10px">${msg}</span>
</li>
```

用户名或者密码为空不能注册

```
$("form").on("submit",function(){

            if($("#name").val() && $("#password").val()){
                return false;
            }

            alert("用户名或密码不能为空");
            return false;
        });
```

用户名已经存在也不能注册

```
$(function(){

        // true-用户名可用
        // false-用户名被占用
        var flag = false;
        //console.log($("#name"));
        $("#name").on("blur",function(){

            var name = $("#name").val();
            if(name && name.trim()){
                $.ajax({
                    type: "get",
                    url: "checkNameServlet",
                    data: "name="+$("#name").val(),
                    dataType: "text",
                    success: function(result){

                        if("用户名可用" == result.trim()){
                            flag = true;
                        }else{
                            flag = false;
                        }
                        $("#show-result").html(result);
                    }
                });
            }

        });

        //console.log($("form"));
        $("form").on("submit",function(){

            if($("#name").val() && $("#password").val()){
                if(flag){
                    return true
                }
                alert("用户名不可用，就先不提交了");
                return false;
            }

            alert("用户名或密码不能为空");
            return false;
        });

    });
```

CustomerService

```
@Override
    public Customer findCustomerByName(String name) {

        SqlSession sqlSession = MyBatisSqlSessionFactory.openSession();
        customerMapper = sqlSession.getMapper(CustomerMapper.class);
        return customerMapper.findCustomerByName(name);
    }
```

md5加密

概述

MD5 是什么

全称为 **消息摘要算法版本5** （Message Digest Algorithm 5）

它是一种 Hash 算法。

作用是为了信息安全。

再具体点，MD5 值就是一串 128 bit 的数据。

MD5 的核心是通过算法把任意长度的原始数据映射成128 bit 的数据。这一点跟CRC 类似，都是把一串数据经过处理，得到另一个固定长度的数据。（所以讲完 CRC，我要讲 MD5, 因为它们两个在实际中经常会出现和用到。）

MD5 的特点：

不可逆性 --- 根据 MD5 值计算不出原始数据

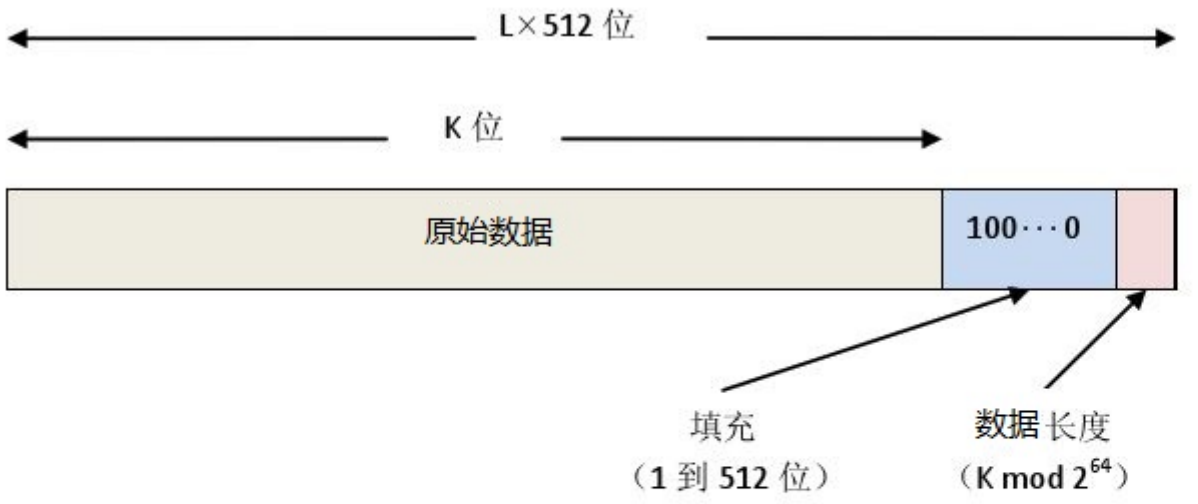
唯一性 --- 不同原始数据会有不同的 MD5 值 （不完全可靠，后面说）

MD5 到底算不算加密算法？仁者见仁智者见智吧。说它是加密，因为它确实把原始数据，比如用户密码，变成了一般人看不懂的 MD5 值；说它不是加密，因为它不能解密。

据说 Linux 系统中，用户密码，都是以 MD5 形式存在文件中的，这样你在输入密码的时候，计算机只要计算你输入密码的 MD5 再跟计算机文件中存储的 MD5 进行比对就行了。

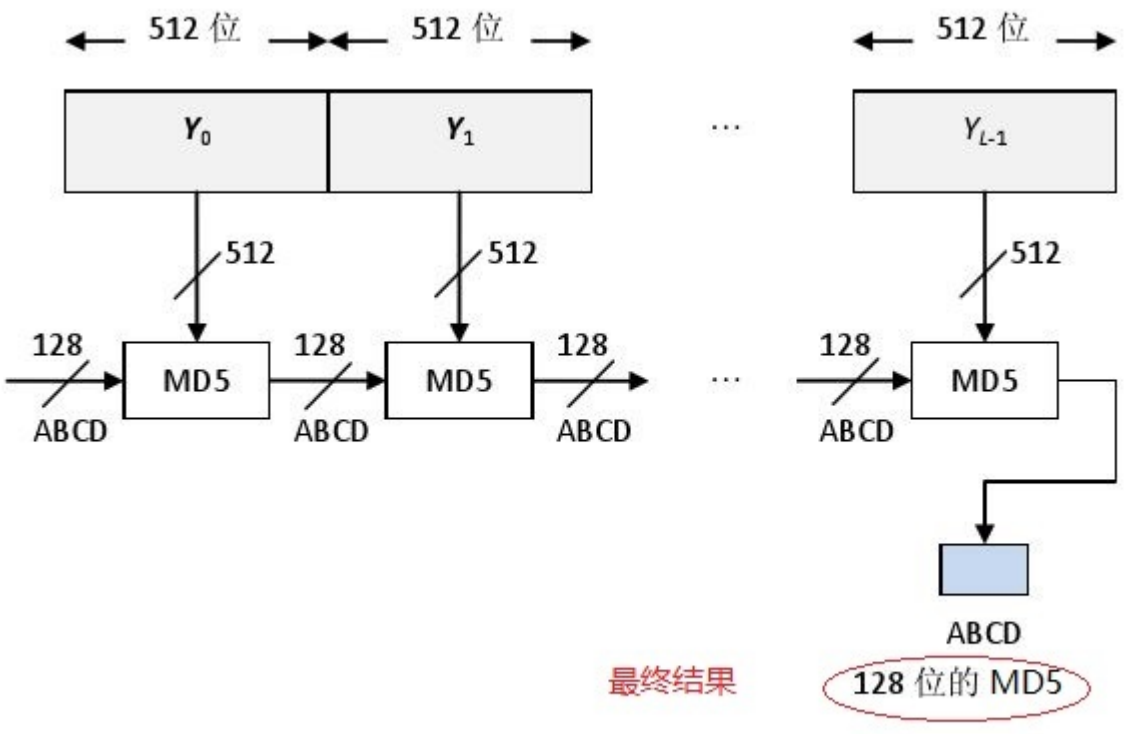
md5算法

第一步：数据填充。填充后数据长度为 512 bit 的整数倍。



注：K 为填充前的数据长度。

第二步： 循环计算。每 512 bit 作为一组，前一个分组得到的 MD5 作为下一个 分组的状态输入 （看作就是ABCD 吧，代表 4 个 32 bit）。最终的ABCD 就是128 bit 的 MD5。



具体的计算过程比较复杂，这里不细说了，说多了大家看起来比较吃力。上面两幅图已经说明了大体的流程。

hash碰撞

Hash 碰撞是指两份不同的原始数据，得到相同的 MD5 值。

我前面已经提到，MD5 是具有唯一性的，其实---这个唯一性是有限的，有概率的。

MD5 之所以应用这么广泛，就是因为它的可靠性，很难有两个不同的输入，得到相同的 MD5。但是！！！虽然概率低 （具体有多大概率，我还没研究清楚），但是确实有。

2004 年山东大学的王晓云就破解了 MD5， 找到了 Hash 碰撞。

其实，Hash 碰撞还是小概率事件了，要不然为何如今尽管已经有人破解了 MD5, 但它还是被大家屁颠屁颠的用着呢?

实现

```
package com.briup.estore.utils;

import java.math.BigInteger;
import java.security.MessageDigest;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/11-06-11-16:44
 * @Description: MD5加密算法
 */
public class MD5Util {
    public static String encode(String str){
        try {
            // 构建MD5实例
            /*
                <li>{@code MD5}</li>
             * <li>{@code SHA-1}</li>
             * <li>{@code SHA-256}</li>
             */
            MessageDigest md = MessageDigest.getInstance("MD5");
            // 产生加密字节数组
            byte[] digest = md.digest(str.getBytes("UTF-8"));
            // 将BigInteger的符号大小表示形式转换为BigInteger。
            return new BigInteger(1,digest).toString(16).toUpperCase();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public static void main(String[] args) {
        // 5FA4D6FC78072F42E0B9817D310BCD35
        System.out.println(encode("briup"));
    }
```

```
}
```

整合

```
 // MD5加密 注册
        customer.setPassword(MD5Util.md5(password));
        customerMapper.saveCustomer(customer);

// 将页面的密码加密后与数据库密文匹配 登录
        if(!MD5Util.md5(password).equals((customer.getPassword()))){
            throw new RuntimeException("密码有误");
        }
```

书籍模块

查看所有目录

分析

- 访问首页 即可查看所有数据
- 此时数据应该随应用启动便查找(监听器)
- 一级目录
  - `select * from es_category where parent_id = 0`
- 二级目录
  - `select * from es_category where parent_id =(select id from es_category where parent_id = 0)`

ApplicationListener

注意: @WebListener

```
package com.briup.web.listener;

import com.briup.bean.Category;
import com.briup.mapper.CategoryMapper;
```

```java
import com.briup.util.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;
import java.util.List;

/**
 * @Auther: vanse
 * @Date: 2021/9/10-09-10-9:50
 * @Description: 项目启动 准备目录和书籍数据
 * @version: 1.0
 */
@WebListener
public class ApplicationListener implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        // 项目启动初始化操作
        // 查询所有目录
        SqlSession session = MyBatisSqlSessionFactory.openSession();
        CategoryMapper categoryMapper = session.getMapper(CategoryMapper.class);
        List<Category> categoryList = categoryMapper.findAllCategory();
        ServletContext servletContext = sce.getServletContext();
        servletContext.setAttribute("categoryList",categoryList);

    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {

    }
}
```

此时使用监听器有缺陷 因为服务器只会加载一次

CategoryMapper.xml

注意: 二级目录的parent_id为一级目录的id

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.briup.estore.dao.CategoryDao">
    <resultMap id="categoryMap" type="category">
        <!--一级目录封装-->
```

```
        <id property="id" column="categoryid"/>
        <result property="name" column="name"/>
        <result property="description" column="description"/>
        <!--一对多属性(二级分类) select 调用sql-->
        <collection property="categories" select="secondeCategoryList"
column="categoryid"/>
    </resultMap>
    <!--一级分类-->
    <select id="findAllCategoryList" resultMap="categoryMap">
        SELECT categoryid,name,description from ES_CATEGORY where PARENT_ID is null
    </select>
    <!--二级分类-->
    <select id="secondeCategoryList" resultMap="categoryMap" parameterType="int">
        SELECT categoryid,name,description FROM ES_CATEGORY
        where PARENT_ID = #{id}
    </select>

</mapper>
```

mybatis-config.xml

```
 <!-- 配置映射文件的位置 -->
    <mappers>
        <mapper resource="mappers/CustomerMapper.xml"/>
        <mapper resource="mappers/CategoryMapper.xml"/>
    </mappers>
```

index.jsp

```
 <dl>
                <%--
                    items 获取范围对象中的数据
                    var   数据集合的某一项    foeach(Category category : List<Category>)
{}
                --%>
                <c:forEach items="${categoryList}" var="firstCa">
                    <dd>
                        <h1>${firstCa.name}</h1>
                        <p>
                            <c:forEach items="${firstCa.categories}" var="secondCa">
                                <a href="list.html">${secondCa.name}</a>
                            </c:forEach>
                        </p>
                    </dd>
                </c:forEach>

            </dl>
```

查看所有书

分析

- 访问首页 即可查看所有数据
- 需要查询目录信息
- 此时数据应该随应用启动便查找(监听器)

ApplicationListener

```
/**
 * @Auther: vanse
 * @Date: 2021/9/10-09-10-9:50
 * @Description: 项目启动 准备目录和书籍数据
 * @version: 1.0
 */
@WebListener
public class ApplicationListener implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        // 项目启动初始化操作
        // 查询所有目录
        SqlSession session = MyBatisSqlSessionFactory.openSession();
        CategoryMapper categoryMapper = session.getMapper(CategoryMapper.class);
        List<Category> categoryList = categoryMapper.findAllCategory();
        ServletContext servletContext = sce.getServletContext();
        servletContext.setAttribute("categoryList",categoryList);

        // 查询所有书籍
        BookMapper bookMapper = session.getMapper(BookMapper.class);
        List<Book> bookList = bookMapper.findAllBook();
        servletContext.setAttribute("bookList",bookList);

    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {

    }
}
```

BookMapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.briup.estore.dao.BookDao">

    <resultMap id="bookMap" type="book">
        <id property="id" column="bookid"/>
        <result property="name" column="bname"/>
        <result property="price" column="price"/>
        <result property="author" column="author"/>
        <result property="publisher" column="publisher"/>
        <result property="pubDate" column="pubdate"/>
        <result property="description" column="bdesc"/>
        <result property="image" column="image"/>
        <!--多对一-->
        <association property="category"
resultMap="com.briup.estore.dao.CategoryDao.categoryMap"/>
    </resultMap>
    <select id="findAllBook" resultMap="bookMap">
        SELECT
            b.bookid,b.name
bname,b.price,b.author,b.publisher,b.pubdate,b.description
bdesc,b.image,b.category_id,
            c.categoryid,c.name,c.description
        FROM
            es_book b,
            es_category c
        where b.category_id=c.categoryid
    </select>
</mapper>
```

mybatis-config.xml

```
<mapper resource="mappers/BookMapper.xml"/>
```

index.jsp

```
 <c:forEach items="${bookList}" var="book">
                <li class="no_mr">
                    <div class="c3_b2_txt">
                        <h1>${book.name}</h1>
                        <p>${book.author}</p>
                        <h2>${book.price}</h2>
                        <h2>${book.category.name}</h2>
                        <p><a href="viewBook.html">更多精彩，点击进入</a></p>
                    </div>
```

```
                    <div style="float: right;position: relative;top: -170px;left:
-48px">
                        <%--图片标签--%>
                        <img src="${book.image}" alt="图片路径有误" width="100px"
height="100px">
                    </div>
                </li>
            </c:forEach>
```

热门书籍(todo)

IndexServlet

```
package com.briup.estore.web.listener;

import com.briup.estore.entity.Book;
import com.briup.estore.entity.Category;
import com.briup.estore.service.IBookService;
import com.briup.estore.service.ICategoryService;
import com.briup.estore.service.impl.BookServiceImpl;
import com.briup.estore.service.impl.CategoryServiceImpl;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;
import java.util.*;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/1-06-01-10:36
 * @Description: 首页监听器: 准备目录和书籍数据
 */
@WebListener
public class IndexListener implements ServletContextListener {
    ICategoryService categoryService = new CategoryServiceImpl();
    IBookService bookService = new BookServiceImpl();
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        ServletContext servletContext = sce.getServletContext();
        List<Category> allCategory = categoryService.findAllCategory();
        System.out.println(allCategory);
        List<Book> allBook = bookService.findAllBook();
        // 存目录
        servletContext.setAttribute("categoryList",allCategory);
        // 存书籍
        servletContext.setAttribute("bookList",allBook);
        // 存热门书籍 (随机三本书)
```

```
        Set<Book> hostBook = new HashSet<>();
        Random random = new Random();
        while (hostBook.size() < 3){
            // 随机生成索引
            int index = random.nextInt(allBook.size());
            hostBook.add(allBook.get(index));
        }
        servletContext.setAttribute("hostBookList",hostBook);
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {

    }
}
```

index.jsp

```
<ul>
    <c:forEach items="${hostBookList}" var="host">
        <li>
            <a href="viewBook.html"><img src="${host.image}" width="100px"
height="123px"/></a>
            <a href="viewBook.html">${host.name}</a>
        </li>
    </c:forEach>
</ul>
```

查看书详情

分析

- 页面传递书籍id
- 将id和application范围对象的数据比对即可

ApplicationInit

```java
package com.briup.estore.web.listener;

import com.briup.estore.dao.BookDao;
import com.briup.estore.dao.CategoryDao;
import com.briup.estore.entity.Book;
import com.briup.estore.entity.Category;
import com.briup.estore.utils.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/27-06-27-16:44
 * @Description: 服务器启动 准备首页数据
 */
@WebListener
public class IndexListener implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        // 范围对象    reqeust -> session -> application
        ServletContext application = sce.getServletContext();
        // 项目启动 自动执行代码
        // 查一级目录和二级目录
        SqlSession session = MyBatisSqlSessionFactory.openSession();

        CategoryDao categoryDao = session.getMapper(CategoryDao.class);
        List<Category> categoryList = categoryDao.findAllCategoryList();
        application.setAttribute("categoryList", categoryList);

        // 查书和所在分类
        BookDao bookDao = session.getMapper(BookDao.class);
        List<Book> bookList = bookDao.findAllBook();
        application.setAttribute("bookList", bookList);
        //bookList.forEach(b-> System.out.println(bookList));
        //bookList.forEach(System.out::println);

        // 对方能通过id 获取到book
        Map<Integer,Book> bookMap = new HashMap<>();
        for (Book book : bookList) {
            // 1, {1,"沉思录"...}
            bookMap.put(book.getId(),book);
        }
```

```
            application.setAttribute("bookMap",bookMap);
        }

        @Override
        public void contextDestroyed(ServletContextEvent sce) {
            // 项目关闭 自动执行代码
        }
}
```

index.jsp

```
<c:forEach items="${bookList}" var="book">
    <li class="no_mr">
        <div class="c3_b2_txt">
            <h1>${book.name}</h1>
            <p>${book.author}</p>
            <h2>${book.price}</h2>
            <h2>${book.category.name}</h2>
            <p><a href="/bookInfo?id=${book.id}">更多精彩，点击进入</a></p>
        </div>
        <div style="float: right;position: relative;top: -170px;left: -48px">
            <%--图片标签--%>
            <a href="/bookInfo?id=${book.id}"><img src="${book.image}" alt="图片路径
有误" width="100px" height="100px"></a>
        </div>
    </li>
</c:forEach>
```

BookInfoServlet

```
package com.briup.estore.web.servlet;

import com.briup.estore.entity.Book;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Map;
```

```java
/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/28-06-28-9:54
 * @Description: 书详情 api
 */
@WebServlet("/bookInfo")
public class BookInfoServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 1.接收书的id "5" -> 5
        String id = req.getParameter("id");
        int bookId = Integer.parseInt(id);
        // 2.数据库查该id对应的书
            // select * from es_book where id = #{id}
        ServletContext application = req.getServletContext();
        Map<Integer, Book> bookMap = (Map<Integer, Book>)
application.getAttribute("bookMap");
        Book book = bookMap.get(bookId);
        // 3.存入session
        req.getSession().setAttribute("book",book);
        // 4.跳转到详情页面
        resp.sendRedirect("bookInfo.jsp");
    }
}
```

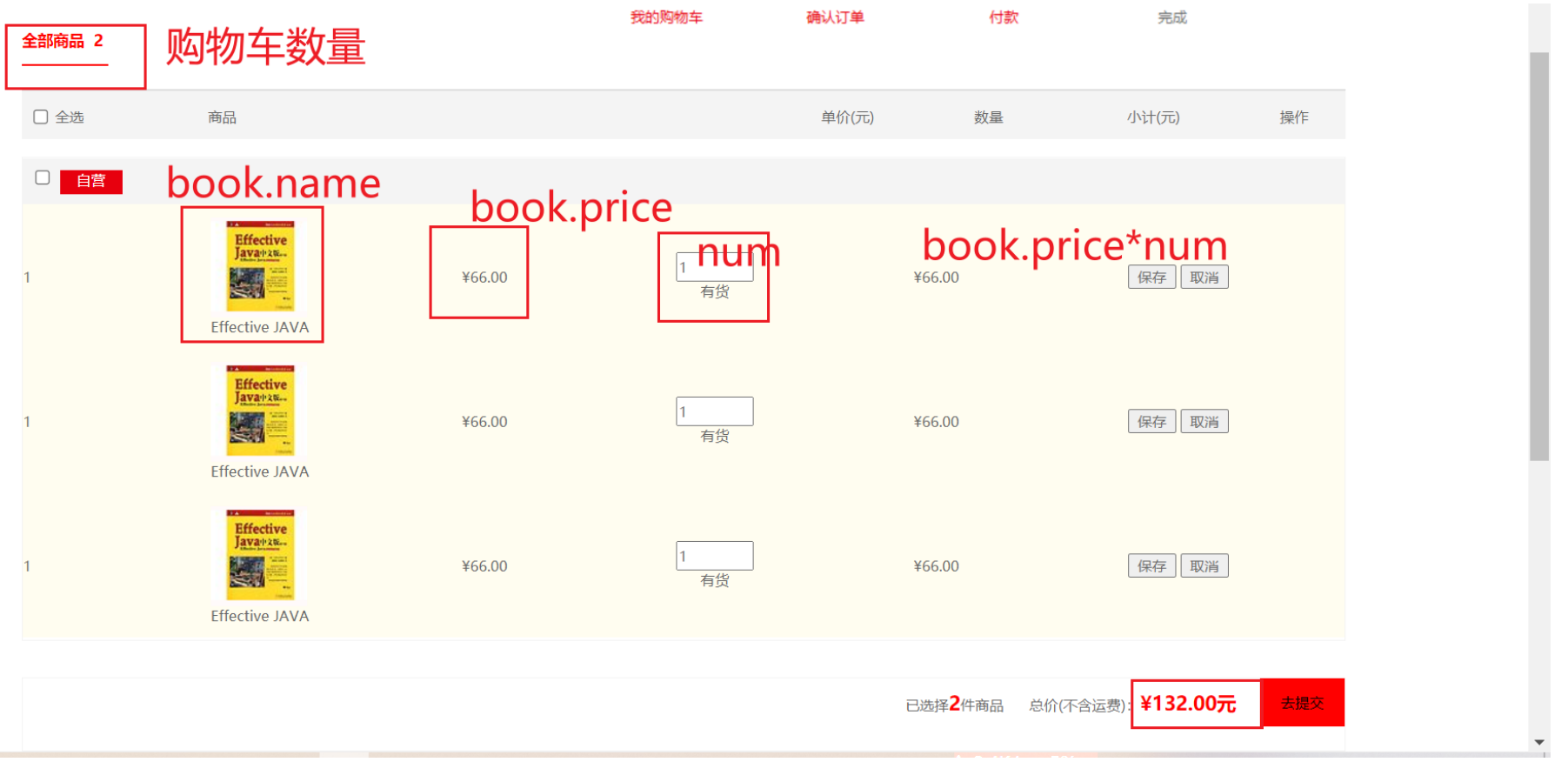
viewBook.jsp

fmt 解决时间格式

```jsp
        <div class="cn5_top_y center01">
            <div class="cn5topy_1">
                <div class="cn5topy_imgview">
                    <img src="${book.image }"/>
                </div>
            </div>
            <div class="cn5topy_2">
                <h1 class="pro_title font15">${book.name }</h1>
                <div class="pro_price">
                    <div class="pro_price_x">
                        <p> briup价：<b>￥${book.price }</b> <a href="#">(降价通知)</a>
</p>
                    </div>

                </div>
                <div class="pro_selects">
                    <div class="pro_select">
```

```
                        <div class="pro_key pro_key_vertical fl">作者：</div>
                        <ul class="pro_select_vals">
                            <li>${book.author }</li>
                        </ul>
                    </div>
                    <div class="pro_select">
                        <div class="pro_key pro_key_vertical fl">出版社：</div>
                        <ul class="pro_select_vals">
                            <li>${book.publisher }</li>

                        </ul>
                    </div>
                    <div class="pro_select">
                        <div class="pro_key pro_key_vertical fl">书籍介绍：</div>
                        <ul class="pro_select_vals">

                            <li>出版时间：<f:formatDate value="${book.pubDate }"
pattern="yyyy-MM-dd"/> </li>
                            <li>描述：${book.description }</li>
                        </ul>
                    </div>
                </div>
                <form action="user/addShopCarServlet" id="addCar" method="post">
                    <div class="pro_buy">
                        <div class="pro_buy_nums">
                            <input type="hidden" name="bookId" value="${book.id }" >
                            <input type="text" value="1" name="num"/>
                            <dl>
                                <dd onclick="addCount()">+</dd>
                                <dd onclick="reduceCount()">-</dd>
                            </dl>
                        </div>
                        <!--  <div class="pro_addshop" onclick="submitForm()">加入购
物车</div> -->
                        <div class="pro_addshop"><a href="javascript:submitForm()">
加入购物车</a></div>
                    </div>
                </form>
                <div class="pro_ship">
                    <div>
                        <div class="pro_key fl">服
      务：</div>
                        <ul class="pro_service f1">
                            <li class="service_fq">分期付款</li>
                            <li class="service_myf">免运费</li>
                            <li class="service_zt">自提</li>
                            <li class="service_th">7天无理由退换货</li>
                        </ul>
                    </div>
                </div>
            </div>
        </div>
```

购物车模块



ShopCart

```
create table es_shopcar(
    id NUMBER,
    num NUMBER,
    book_id NUMBER,
    customer_id NUMBER
)
```

```
package com.briup.estore.entity;
/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/26-06-26-23:25
 * @Description:购物车对象
 */
public class ShopCart {
    private int id;
    private int num;
    private int bookId;
    private int customerId;

    private Customer customer;
    private Book book;
```

```
public ShopCart(){}

public ShopCart(int num, int bookId, int customerId) {
    this.num = num;
    this.bookId = bookId;
    this.customerId = customerId;
}

public int getBookId() {
    return bookId;
}

public void setBookId(int bookId) {
    this.bookId = bookId;
}

public int getCustomerId() {
    return customerId;
}

public void setCustomerId(int customerId) {
    this.customerId = customerId;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public int getNum() {
    return num;
}

public void setNum(int num) {
    this.num = num;
}

public Book getBook() {
    return book;
}

public void setBook(Book book) {
    this.book = book;
}

public Customer getCustomer() {
    return customer;
}
```

```
    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    @Override
    public String toString() {
        return "ShopCart{" +
                "id=" + id +
                ", num=" + num +
                ", bookId=" + bookId +
                ", customerId=" + customerId +
                ", customer=" + customer +
                ", book=" + book +
                '}';
    }
}
```

加入购物车

ShopCarAddServlet

> 注意需要先登录 才能加入购物车

```
package com.briup.estore.web.servlet;

import com.briup.estore.entity.Book;
import com.briup.estore.entity.Customer;
import com.briup.estore.entity.ShopCar;
import com.briup.estore.entity.ShopCart;
import com.briup.estore.service.IShopCarService;
import com.briup.estore.service.impl.ShopCarServiceImpl;
import com.briup.estore.utils.BookUtil;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * @Auther: vanse(lc))
```

```
 * @Date: 2022/6/1-06-01-16:48
 * @Description: 添加购物车api
 */
@WebServlet("/shopCarAdd")
public class ShopCarAddServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 1.获取入库信息
        int bookId = Integer.parseInt(req.getParameter("bookId"));
        int num = Integer.parseInt(req.getParameter("num"));
        HttpSession session = req.getSession();
        Customer customer = (Customer) session.getAttribute("customer");
        // 2.入库
        IShopCarService shopCarService = new ShopCarServiceImpl();
        ShopCart shopCart = new ShopCart(num,bookId,customer.getId());
        shopCarService.add(shopCart);
        // 4.跳转购物车页面
        resp.sendRedirect("shopCartQuery");
    }
}
```

ShopCarService

```
public interface IShopCarService {
    void add(ShopCart shopCart);
    List<ShopCart> getCarBookByCid(Integer id);
}
```

```
package com.briup.estore.service.impl;

import com.briup.estore.dao.IShopCarDao;
import com.briup.estore.entity.Book;
import com.briup.estore.entity.ShopCar;
import com.briup.estore.entity.ShopCart;
import com.briup.estore.service.IShopCarService;
import com.briup.estore.utils.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

import java.util.List;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/26-06-26-23:24
 * @Description: 添加购物车逻辑
 */
public class ShopCarServiceImpl implements IShopCarService {
    SqlSession session = MyBatisSqlSessionFactory.openSession();
    IShopCarDao shopCarDao =  session.getMapper(IShopCarDao.class);
```

```
    @Override
    public void add(ShopCart shopCart) {

        // 是否为同一本书
            // 是: 数量+1
            // 否: 添加新行
        ShopCart cartFromDB = shopCarDao.queryByBId(shopCart.getBookId());
        if (cartFromDB != null) {
            // 同一本书
            cartFromDB.setNum(cartFromDB.getNum()+shopCart.getNum());
            shopCarDao.update(cartFromDB);
        }else{
            // 新书
            shopCarDao.add(shopCart);
        }
        session.commit();
    }

    @Override
    public List<ShopCart> getCarBookByCid(Integer customerId) {
        return shopCarDao.queryByCid(customerId);
    }
}
```

ShopCarDao

```
public interface IShopCarDao {
    void add(ShopCart shopCart);
    ShopCart queryByBId(Integer id);
    void update(ShopCart cartFromDB);
    List<ShopCart> queryByCid(Integer customerId);
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.briup.estore.dao.IShopCarDao">
    <insert id="add">
        insert into es_shopcar(num,book_id,customer_id)
        values (#{num},#{bookId},#{customerId})
    </insert>
    <update id="update">
        update es_shopcar set num = #{num} where id = #{id}
    </update>
    <select id="queryByBId" resultType="com.briup.estore.entity.ShopCart"
parameterType="int">
        select id,num,book_id,customer_id from es_shopcar
        where book_id = #{id}
```

```
    </select>
    <resultMap id="carMap" type="shopCart">
        <id property="id" column="id"/>
        <result property="num" column="num"/>
        <result property="bookId" column="book_id"/>
        <association property="book"
resultMap="com.briup.estore.dao.IBookDao.bookMap"/>
    </resultMap>
    <select id="queryByCid" resultMap="carMap">
         select
            car.id,car.num,car.book_id,
            b.price,b.name,b.image
        from es_book b,es_shopcar car
        where b.id = car.book_id and customer_id = #{customerId}
    </select>


</mapper>
```

shopCart.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false"
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
    <title>购物车页面</title>
    <link rel="stylesheet" href="css/fullCar.css"/>
    <link rel="stylesheet" href="css/common.css"/>
    <link rel="stylesheet" href="css/style.css"/>
    <link rel="stylesheet" href="css/icons.css"/>
    <link rel="stylesheet" href="css/table.css"/>
    <link rel="stylesheet" type="text/css" href="css/shopCar.css">
    <script type="text/javascript" src="js/jquery-1.12.3.min.js"></script>
    <script type="text/javascript">
        function reduceCount() {
            var numObj = document.getElementsByName("num")[0];
            if (parseInt(numObj.value) > 1) {
                numObj.value = parseInt(numObj.value) - 1;
            }
        }

        function addCount() {
            var numObj = document.getElementsByName("num")[0];
            numObj.value = parseInt(numObj.value) + 1;
```

```
        }
    </script>
</head>
<body>
<!--顶部-->
<div class="top">
    <div class="top_center">
        <ul class="top_bars">
            <li><a href="/userLogout">退出</a>|</li>
            <li><a href="#">我的订单<span class="jt_down"></span></a>|</li>
            <li><a href="#">关注杰普<span class="jt_down"></span></a>|</li>
            <li><a href="#">网站导航<span class="jt_down"></span></a></li>
        </ul>
    </div>
</div>
<!--头部-->
<div class="header3">
    <a href="#"><img src="images/logo.png" class="oneImg"></a>


    <div class="h3_right">
        <img src="images/play_03.png" alt="">
    </div>

</div>
<!--中间部分div-->
<div class="empty">
    <div class="peisong">
        <pre>全部商品  ${map.size} </pre>
    </div>
    <div class="mainCenter">
        <div class="allCheck">
            <input type="checkbox">
            <p>全选</p>
            <p class="leftM">商品</p>
            <p class="rightM">单价(元)</p>
            <p class="leftM">数量</p>
            <p class="leftM">小计(元)</p>
            <p class="leftM">操作</p>
        </div>
        <div class="mainPro">
            <div class="aa">
                <input type="checkbox"><span id="but">自营</span>
            </div>
            <div class="cartForm">
                <c:set var="totalCost" value="0"/>
                <c:forEach items="${map.shopCarList}" var="shopCar" varStatus="s">
                    <form action="/shopCarUpdate" method="get">
                        <table>
                            <tbody>
                            <tr>
```

```
                                    <td class="index" style="padding-left:
15px">${s.index+1}</td>
                                    <td>
                                        <img src="${shopCar.book.image}" width="100px"
height="123px">
                                        <span>
                                       ${shopCar.book.name}
                                        <br>
                                    </span>
                                    </td>
                                    <td>
                                        <span>¥${shopCar.book.price}</span>
                                    </td>
                                    <td class="index2">
                                        <input type="text" name="num"
value="${shopCar.num}"><br>
                                            <%--隐藏bookId--%>
                                        <input type="hidden" name="bookId"
value="${shopCar.book.id}"><br>
                                            <%--<dl>
                                                <dd onclick="addCount()">+</dd>
                                                <dd onclick="reduceCount()">-</dd>
                                            </dl>--%>
                                        <span>有货</span>
                                    </td>
                                    <td>
                                        <span>¥${shopCar.book.price * shopCar.num}
</span>
                                    </td>
                                    <c:set var="totalCost" value="${totalCost +
shopCar.book.price * shopCar.num}"/>
                                    <td>
                                        <button type="submit">更新</button>
                                        <button type="button"
 onclick="window.location.href='shopCarRemove?bookId='+${orderLine.book.id}">移除
                                        </button>
                                    </td>
                                </tr>
                                </tbody>
                            </table>
                        </form>
                    </c:forEach>
                </div>

            </div>
            <div class="allButtom">
                <!-- <p class="caozuo">去结算</p> -->
                <!-- <input value="去提交" class="caozuo" type="submit"> -->
                <a href="/orderConfirm" class="caozuo">去提交</a>
                <span>已选择<font>${map.size}</font>件商品
      总价(不含运费)：<font>¥${totalCost}元</font></span>
```

```
        </div>

    </div>

</div>


<!--脚部-->
<div class="footer3">
    <div class="f3_top">
        <div class="f3_center">
            <div class="ts1">品目繁多 愉悦购物</div>
            <div class="ts2">正品保障 天天低价</div>
            <div class="ts3">极速物流 特色定制</div>
            <div class="ts4">优质服务 以客为尊</div>
        </div>
    </div>
    <div class="f3_middle">
        <ul class="f3_center">
            <li class="f3_mi_li01">
                <h1>购物指南</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li01">
                <h1>配送方式</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li01">
                <h1>支付方式</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li01">
                <h1>售后服务</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li01">
                <h1>服务保障</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
```

```
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li06">
                <h1>客服中心</h1>
                <img src="images/qrcode_jprj.jpg" width="80px" height="80px">
                <p>抢红包、疑问咨询、优惠活动</p>
            </li>
        </ul>
    </div>
    <div class="f3_bottom">
        <p class="f3_links">
            <a href="#">关于我们</a>|
            <a href="#">联系我们</a>|
            <a href="#">友情链接</a>|
            <a href="#">供货商入驻</a>
        </p>
        <p>沪ICP备14033591号-8 杰普软件briup.com版权所有 杰普软件科技有限公司 </p>
        <img src="images/police.png">
    </div>
</div>
</body>
</html>
```

更新购物车数量(todo)

- form中的button按钮会提交表单
    - 给定num值和bookId值(隐藏框)
- 传递num和id给后台
- 后台根据用户id,bookId,num 修改购物车数量

移除购物车(todo)

- 点击移除按钮 传递购物车id
- servlet接收id
- dao层根据id删除购物车
- 重新跳转shopQuery
    - 此时查的数据少一条

订单模块

准备提交

分析

- 在真正提交之前,先跳转到确认页面
- 选择收货信息
  - 查找以前的收货信息
  - 添加新的收货信息

ConfirmServlet

```
package com.briup.estore.web.servlet;

import com.briup.estore.entity.Customer;
import com.briup.estore.entity.ShopAddress;
import com.briup.estore.service.IAddressService;
import com.briup.estore.service.impl.AddressServiceImpl;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.List;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/2-06-02-11:27
 * @Description: com.briup.estore.web.servlet
 */
@WebServlet("/orderConfirm")
public class OrderConfirmServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 1.准备用户的收获地址
        HttpSession session = req.getSession();
        // 当前登录用户
        Customer customer = (Customer) session.getAttribute("customer");
        IAddressService adressService = new AddressServiceImpl();
        List<ShopAddress> shopAddressList =
adressService.findAddressByCId(customer.getId());
        session.setAttribute("shopAddressList",shopAddressList);
        // 2.跳转确认页面
        resp.sendRedirect("orderConfirm.jsp");
    }
```

```
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        doGet(req,resp);
    }
}
```

ShopAddressServiceImpl

```
package com.briup.service.impl;

import com.briup.bean.ShopAddress;
import com.briup.dao.ShopAddressMapper;
import com.briup.service.ShopAddressService;
import com.briup.util.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

import java.util.List;

/**
 * @Auther: vanse
 * @Date: 2021/9/10-09-10-13:45
 * @Description: com.briup.service.impl
 * @version: 1.0
 */
public class ShopAddressServiceImpl implements ShopAddressService {
    private SqlSession sqlSession = MyBatisSqlSessionFactory.openSession(true);
    private ShopAddressMapper shopAddressMapper =
sqlSession.getMapper(ShopAddressMapper.class);

    @Override
    public List<ShopAddress> findAddressByCustomer(Integer id) {
        return shopAddressMapper.findShopAddressByCId(id);
    }
}
```

####ShopAddressMapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.briup.dao.ShopAddressMapper">
<select id="findShopAddressByCId" parameterType="int"
    resultType="com.briup.bean.ShopAddress">
        select *
        from es_shopaddress
        where customer_id = #{id}
    </select>
</mapper>
```

confirm.jsp

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

<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false"
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
    <title>结算页面</title>
    <link rel="stylesheet" href="css/common.css"/>
    <link rel="stylesheet" href="css/icons.css" />
    <link rel="stylesheet" href="css/table.css" />
    <link rel="stylesheet" href="css/orderSure.css" />
    <script type="text/javascript">

        function showAdres(sp){
            var pa = document.getElementById("newAdres");
            console.log(pa.style['display']);
            if(pa.style['display']=="none"){
                pa.style['display'] = 'block';
                sp.innerHTML = "取消新增地址";
            }else{
                pa.style['display'] = 'none';
                sp.innerHTML = "新增收货地址";
            }
        }
    </script>
</head>
<body>
<!--顶部-->
<div class="top">
    <div class="top_center">
        <ul class="top_bars">
```

```
            <li><a href="#">退出</a>|</li>
            <li><a href="#">我的订单<span class="jt_down"></span></a>|</li>
            <li><a href="#">关注杰普<span class="jt_down"></span></a>|</li>
            <li><a href="#">网站导航<span class="jt_down"></span></a></li>
        </ul>
    </div>
</div>
<!--头部-->
<div class="header3">
    <a href="#"><img src="images/logo.png"  class="oneImg"></a>


    <div class="h3_right">
        <img src="images/play_03.png" alt="">
    </div>

</div>
<p class="orderButtom">填写并核对订单信息</p>
<!--中间复杂部分-->
<div class="content">
    <div class="contentCenter">

        <form action="/submit" name="orderForm">

            <div class="centerTop">
                <b style="font-size:20px;">收货人信息</b>

                <b style="float: right;cursor: pointer;" onclick="showAdres(this);"
                   ondblclick="hideAdres(this);">新增收货地址</b>
                <ul class="adres">
                    <c:forEach items="${shopAddressList}" var="shopAddrss">
                        <li>
                            <input type="radio" name="shopAddId"
value="${shopAddrss.id}" style="width:50px">
                            ${shopAddrss.receiveName}   
${shopAddrss.address}    ${shopAddrss.phone}
                        </li>
                    </c:forEach>

                </ul>

                <p id="newAdres" style="font-size:15px;display: none;">
                    收件人姓名：<input type="text" name="receiveName"><br/><br/>
                    收件人电话：<input type="text" name="phone" >
                    收件人地址：<input type="text" name="address"><br/><br/>
                </p>
            </div>



            <p class="singleP"><b>送货清单</b><span><a href="shopCart.jsp">返回修改购物车
</a></span></p>
```

```
            <div class="bigDiv">
                <div class="twoDiv">
                    <b>商家：briup自营</b>
                    <ul class="oneUL">
                        <c:forEach items="${shopCart.orderLines}" var="orderLine">
                            <li>
                                <img src="${orderLine.book.image}" class="pic">
                                <p>计算机
  ${orderLine.book.category.name}  ${orderLine.book.name} &
nbsp;</p>
                                <p><font>¥${orderLine.book.price}
</font>     ×${orderLine.num}     
     有货</p>
                                <p><img src="images/sureLogo_14.png" alt=""><span>七
天无理由退换货</span></p>

                            </li>
                        </c:forEach>

                        <li><pre>【赠品】   高级定制干花书签   随机 ×1</pre></li>
                    </ul>
                </div>
            </div>

            <div class="money">
                <span><font>${shopCart.size}</font>件商品，总商品金额：
              
              
¥${shopCart.totalCost}</span>
                <span><img src="images/sureLogo_18.png" alt="">运费：
              
   ¥0.00</span>

                <span>应付总额：
             ¥
${shopCart.totalCost}</span>
            </div>
            <div class="submit">
                <span>应付金额：<font>¥${shopCart.totalCost}</font><input type="image"
src="images/21_03.png"></span>
            </div>
        </form>
    </div>

</div>
<!--脚部-->
<div class="footer3">
    <div class="f3_top">
        <div class="f3_center">
            <div class="ts1">品目繁多 愉悦购物</div>
            <div class="ts2">正品保障 天天低价</div>
            <div class="ts3">极速物流 特色定制</div>
            <div class="ts4">优质服务 以客为尊</div>
```

```
        </div>
    </div>
    <div class="f3_middle">
        <ul class="f3_center">
            <li class="f3_mi_li01">
                <h1>购物指南</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li01">
                <h1>配送方式</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li01">
                <h1>支付方式</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li01">
                <h1>售后服务</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li01">
                <h1>服务保障</h1>
                <p>常见问题</p>
                <p>找回密码</p>
                <p>会员介绍</p>
                <p>购物演示</p>
            </li>
            <li class="f3_mi_li06">
                <h1>客服中心</h1>
                <img src="images/qrcode_jprj.jpg" width="80px" height="80px">
                <p>抢红包、疑问咨询、优惠活动</p>
            </li>
        </ul>
    </div>
    <div class="f3_bottom">
        <p class="f3_links">
            <a href="#">关于我们</a>|
            <a href="#">联系我们</a>|
            <a href="#">友情链接</a>|
            <a href="#">供货商入驻</a>
```

```
            </p>
            <p>沪ICP备14033591号-8 杰普软件briup.com版权所有 杰普软件科技有限公司 </p>
            <img src="images/police.png">
        </div>
    </div>
    </body>
    </html>
```

提交订单

分析

- 封装订单数据
    - 封装基本信息
    - 封装地址信息
        - 如果选择的是旧地址
        - 如果插入了新地址
    - 封装客户信息
    - 封装订单项信息
- 清空购物车
- 根据客户查找订单信息
- 跳转订单列表

SubmitServlet

注意: 页面输入框一定有值 空字符串

```
package com.briup.web.servlet;

import com.briup.bean.*;
import com.briup.service.OrderFormService;
import com.briup.service.ShopAddressService;
import com.briup.service.impl.OrderFormServiceImpl;
import com.briup.service.impl.ShopAddressServiceImpl;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```java
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.Collection;
import java.util.Date;

/**
 * @Auther: vanse
 * @Date: 2021/9/10-09-10-16:26
 * @Description: 提交订单
 * @version: 1.0
 */
@WebServlet("/submit")
public class SubmitServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        String shopAddId =req.getParameter("shopAddId");
        String receiveName = req.getParameter("receiveName");
        String address = req.getParameter("address");
        String phone = req.getParameter("phone");
        HttpSession session = req.getSession();
        Customer customer = (Customer) session.getAttribute("customer");
        ShopCart shopCart = (ShopCart) session.getAttribute("shopCart");
        ShopAddress shopAddress = null;
        ShopAddressService shopAddressService = new ShopAddressServiceImpl();
        // 获取地址信息
        if (!"".equals(receiveName) && !"".equals(address) && !"".equals(phone)) {
            // 新地址 需要保存到数据库中
            shopAddress = new ShopAddress();
            shopAddress.setCustomer(customer);
            shopAddressService.saveShopAddress(shopAddress);
        } else {
            if(shopAddId != null){
                // 旧地址
                shopAddress =
shopAddressService.findAddressById(Integer.parseInt(shopAddId));
            }

        }
        Collection<OrderLine> orderLines = shopCart.getOrderLines();
        double cost = 0;
        for (OrderLine orderLine : orderLines) {
            cost += orderLine.getCost();
        }
        // 封装订单类
        OrderForm orderForm = new OrderForm();
        orderForm.setShopAddress(shopAddress);
        orderForm.setOrderLines(orderLines);
        orderForm.setCustomer(customer);
        orderForm.setCost(cost);
        orderForm.setOrderDate(new Date());
```

```
            // 添加订单
            OrderFormService orderFormServcie = new OrderFormServiceImpl();
            orderFormServcie.saveOrderForm(orderForm);
            // 清空购物车
            shopCart.clear();
            // 查询订单
            resp.sendRedirect("/findAllOrderForm");

        }
}
```

ShopAddressServiceImpl

```
package com.briup.service.impl;

import com.briup.bean.ShopAddress;
import com.briup.dao.ShopAddressMapper;
import com.briup.service.ShopAddressService;
import com.briup.util.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

import java.util.List;

/**
 * @Auther: vanse
 * @Date: 2021/9/10-09-10-13:45
 * @Description: com.briup.service.impl
 * @version: 1.0
 */
public class ShopAddressServiceImpl implements ShopAddressService {
    private SqlSession sqlSession = MyBatisSqlSessionFactory.openSession(true);
    private ShopAddressMapper shopAddressMapper =
sqlSession.getMapper(ShopAddressMapper.class);

    @Override
    public List<ShopAddress> findAddressByCustomer(Integer id) {
        return shopAddressMapper.findShopAddressByCId(id);
    }

    @Override
    public void saveShopAddress(ShopAddress shopAddress) {
        shopAddressMapper.saveShopAddress(shopAddress);
    }

    @Override
    public ShopAddress findAddressById(int shopAddId) {
        return shopAddressMapper.findAddressById(shopAddId);
    }
```

```
}
```

ShopAddressMapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.briup.dao.ShopAddressMapper">
    <select id="findShopAddressByCId" parameterType="int"
    resultType="com.briup.bean.ShopAddress">
        select *
        from es_shopaddress
        where customer_id = #{id}
    </select>

    <insert id="saveShopAddress" parameterType="shopAddress" useGeneratedKeys="true"
keyProperty="id">
        insert into es_shopaddress(receiveName,address,phone,customer_id)
        values(#{receiveName},#{address},#{phone},#{customer.id})
    </insert>

    <select id="findAddressById" resultType="shopAddress" parameterType="int">
        select id,receiveName,address,phone,customer_id
        from es_shopaddress where id= #{id}
    </select>
</mapper>
```

OrderFormServiceImpl

注意: 订单项需要保存订单id

```
package com.briup.service.impl;

import com.briup.bean.OrderForm;
import com.briup.bean.OrderLine;
import com.briup.dao.OrderFormMapper;
import com.briup.dao.OrderLineMapper;
import com.briup.service.OrderFormService;
import com.briup.util.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

import java.util.Collection;

/**
 * @Auther: vanse
 * @Date: 2021/9/10-09-10-16:51
```

```
 * @Description: com.briup.service.impl
 * @version: 1.0
 */
public class OrderFormServiceImpl implements OrderFormService {
    SqlSession session = MyBatisSqlSessionFactory.openSession(false);
    OrderFormMapper orderFormMapper = session.getMapper(OrderFormMapper.class);
    OrderLineMapper orderLineMapper = session.getMapper(OrderLineMapper.class);
    @Override
    public void saveOrderForm(OrderForm orderForm) {
        try{
            // 保存订单
            orderFormMapper.saveOrderForm(orderForm);
            // 保存订单项
            Collection<OrderLine> orderLines = orderForm.getOrderLines();
            orderLines.forEach(orderLine -> {
                orderLine.setOrderForm(orderForm);
            });
            orderLineMapper.saveOrderLine(orderForm.getOrderLines());
            session.commit();
        }catch (Exception e){
            session.rollback();
            e.printStackTrace();
        }

    }
}
```

OrderFormMapper.java

```
public interface OrderFormMapper {
    void saveOrderForm(OrderForm orderForm);
}
```

OrderFormMapper.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.briup.dao.OrderFormMapper">
    <insert id="saveOrderForm" parameterType="orderForm"
        useGeneratedKeys="true" keyProperty="id">
        insert into
        es_orderform(cost,orderDate,shopAddress_id,customer_id)
        values(#{cost},#{orderDate},#{shopAddress.id},#{customer.id})
    </insert>
</mapper>
```

OrderLineMapper.java

```java
public interface OrderLineMapper {
    void saveOrderLineList(Collection<OrderLine> orderLines);
}
```

OrderLineMapper.xml

此处使用了批量插入 也可单个循环插入

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.briup.dao.OrderLineMapper">
    <insert id="saveOrderLine" parameterType="collection"
            useGeneratedKeys="true" keyProperty="id">

        insert into
        es_orderline(num,cost,book_id,orderForm_id)
        values

        <foreach collection="collection" item="orderLine" index="index"
separator=",">
            (
                #{orderLine.num},
                #{orderLine.cost},
                #{orderLine.book.id},
                #{orderLine.orderForm.id}
            )
        </foreach>
    </insert>
```

```
</mapper>
```

ShopCart

```
public void clear() {
        cart.clear();
    }
```

查看自己订单

分析

- 客户只能查到与自己相关的订单
- 级联地址信息

FindAllOrderForm

```
package com.briup.web.servlet;

import com.briup.bean.Customer;
import com.briup.bean.OrderForm;
import com.briup.service.OrderFormService;
import com.briup.service.impl.OrderFormServiceImpl;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.List;

/**
 * @Auther: vanse
 * @Date: 2021/9/11-09-11-8:41
 * @Description: com.briup.web.servlet
 * @version: 1.0
 */
@WebServlet("/findAllOrderForm")
public class FindAllOrderForm extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        HttpSession session = req.getSession();
        Customer customer = (Customer) session.getAttribute("customer");
```

```
        OrderFormService orderFormService = new OrderFormServiceImpl();
        List<OrderForm> orderFormList =
orderFormService.findAllOrderForm(customer.getId());
        session.setAttribute("orderFormList",orderFormList);
        resp.sendRedirect("orderList.jsp");
    }
}
```

OrderFormServiceImpl

```
@Override
    public List<OrderForm> findAllOrderForm(Integer id) {
        return orderFormMapper.findAllOrderForm(id);
    }
```

OrderFormMapper.xml

```
<resultMap id="orderFormMap" type="orderForm">
        <id column="id" property="id" />
        <result column="cost" property="cost" />
        <result column="orderDate" property="orderDate" />
        <!--收货人 注意: findAddressById的返回形式需要resultMap-->
        <association property="shopAddress"
            column="shopAddress_id"
select="com.briup.dao.ShopAddressMapper.findAddressById"/>
    </resultMap>
    <select id="findAllOrderForm" resultMap="orderFormMap">
        select id,cost,orderDate,shopAddress_id,customer_id
        from
        es_orderform
        where customer_id = #{id}
    </select>
```

ShopAddressMapper.xml

```
<resultMap type="ShopAddress" id="ShopAddressResult">
        <id column="id" property="id" />
        <result column="receiveName" property="receiveName" />
        <result column="address" property="address" />
        <result column="phone" property="phone" />
    </resultMap>
    <select id="findAddressById" resultMap="ShopAddressResult" parameterType="int">
        select id,receiveName,address,phone,customer_id
        from es_shopaddress where id= #{id}
    </select>
```

orderList.jsp

```
<c:forEach items="${orderFormList}" var="orderForm" varStatus="v">
            <tr>
                <td align="center">${v.index+1}</td>
                <td>${orderForm.id}</td>
                <td>${orderForm.cost}</td>
                <td>未付款</td>
                <td>${orderForm.shopAddress.receiveName}</td>
                <td><input type="button" value="删除"
onclick="javascript:window.location='orderlist.html';"><input type="button"
value="明细" onclick="javascript:window.location='orderdetail.html';">  </td>
            </tr>
        </c:forEach>
```

查看订单详情(todo)

分析

- 级联书的信息

####orderList.jsp

```
<input type="button" value="明细" onclick="javascript:window.location='/orderDetail?
id=${orderForm.id}';">
```

OrderDetailServlet

```java
@WebServlet("/orderDetail")
public class OrderDetailServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        int id = Integer.parseInt(req.getParameter("id"));
        OrderLineService orderLineService = new OrderLineServiceImpl();
        List<OrderLine> orderLineList =
orderLineService.findOrderLinesByOrderFormId(id);
        HttpSession session = req.getSession();
        session.setAttribute("orderLineList",orderLineList);
        req.getRequestDispatcher("/orderDetail.jsp").forward(req,resp);
    }
}
```

OrderLineServiceImpl

```java
package com.briup.service.impl;

import com.briup.bean.OrderLine;
import com.briup.dao.OrderLineMapper;
import com.briup.service.OrderLineService;
import com.briup.util.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

import java.util.Collection;
import java.util.List;

/**
 * @Auther: vanse
 * @Date: 2021/9/11-09-11-9:22
 * @Description: com.briup.service.impl
 * @version: 1.0
 */
public class OrderLineServiceImpl implements OrderLineService {
    SqlSession session = MyBatisSqlSessionFactory.openSession(false);
    OrderLineMapper orderLineMapper = session.getMapper(OrderLineMapper.class);
    @Override
    public List<OrderLine> findOrderLinesByOrderFormId(int id) {
        return orderLineMapper.findOrderLinesByOrderFormId(id);
    }

    @Override
    public void saveOrderLine(Collection<OrderLine> orderLines) {
        orderLineMapper.saveOrderLine(orderLines);
    }
}
```

OrderLineMapper.xml

```
<resultMap type="OrderLine" id="OrderLineResult">
        <id column="id" property="id" />
        <result column="num" property="num" />
        <result column="cost" property="cost" />
        <association property="book" column="book_id"
            select="com.briup.dao.BookMapper.findBookById"/>
    </resultMap>
    <select id="findOrderLinesByOrderFormId" resultMap="OrderLineResult">
        select * from es_orderline
        where orderForm_id = #{id}
    </select>
```

BookMapper.xml

```
<resultMap type="book" id="simpleBookMap">
        <id column="id" property="id"/>
        <result column="name" property="name"/>
        <result column="price" property="price"/>
        <result column="author" property="author"/>
        <result column="publisher" property="publisher"/>
        <result column="pubDate" property="pubDate"/>
        <result column="description" property="description"/>
        <result column="image" property="image"/>
    </resultMap>
    <select id="findBookById" parameterType="int" resultMap="simpleBookMap">
        select * from es_book
        where id = #{id}
    </select>
```

orderDetail.jsp

此处使用c:set 循环计算

```
<tbody>
        <c:set var="totalCost" value="0"/>
        <c:forEach items="${orderLineList}" var="orderLine" varStatus="v">
            <tr>
                <td>${v.index}</td>
                <td>${orderLine.book.name}</td>
                <td>价格：<span>${orderLine.book.price}</span></td>
                <td>数量：<span>${orderLine.num}</span></td>
                <td>小计：<span> ${orderLine.book.price} * ${orderLine.num}</span>
</td>
```

```
                <c:set var="totalCost" value="${orderLine.book.price * orderLine.num
+ totalCost}"/>
            </tr>
        </c:forEach>
        <tr>
            <td colspan="5" class="count">合计：<span>${totalCost}</span></td>
        </tr>
        </tbody>
```

删除订单(todo)

分析

- 订单被订单项引用 不能直接删除
- 先删除引用了订单的所有订单项信息,再删除订单

orderList.jsp

```
<input type="button" value="删除" onclick="javascript:window.location='/orderDelete?
id=${orderForm.id}';">
```

OrderDelete

```
package com.briup.web.servlet;

import com.briup.service.OrderFormService;
import com.briup.service.impl.OrderFormServiceImpl;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * @Auther: vanse
 * @Date: 2021/9/11-09-11-11:11
 * @Description: com.briup.web.servlet
 * @version: 1.0
 */
@WebServlet("/orderDelete")
public class OrderDelete extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
```

```
        int id = Integer.parseInt(req.getParameter("id"));
        OrderFormService orderFormService  = new OrderFormServiceImpl();
        orderFormService.deleteOrderById(id);

        req.getRequestDispatcher("/findAllOrderForm").forward(req,resp);
    }
}
```

OrderFormServiceImpl

注意: 订单和订单项有外键关联

```
 @Override
    public void deleteOrderById(int id) {
        List<OrderLine> orderList =
orderLineService.findOrderLinesByOrderFormId(id);
        orderLineService.deleteBatch(orderList);
        orderFormMapper.deleteById(id);
    }
```

OrderLineServiceImpl

```
@Override
public void deleteBatch(List<OrderLine> orderList) {
    orderLineMapper.deleteBatch(orderList);
}
```

OrderLineMapper.xml

```
<delete id="deleteBatch" parameterType="list">
        delete from es_orderLine
        where id in
        <foreach collection="list" open="(" separator="," close=")"
item="orderLine">
            #{orderLine.id}
        </foreach>
   </delete>
```

OrderFormMapper.xml

```
<delete id="deleteById" parameterType="int">
        delete from es_orderform
        where id = #{id}
    </delete>
```

集成沙箱

注册沙箱

注意： 不要使用谷歌浏览器...否则沙箱页面无法刷新

1.登陆蚂蚁金服官网 https://www.antgroup.com/

2.划到尾页，点击支付宝开放平台

3.点击立即入驻（注册过的直接扫码登陆）

4.填写身份信息注册沙箱账号

5.注册完跳转页面，选择沙箱

6.跳转页面，点击开放平台控制台

以正式环境为准。

- 为保证沙箱稳定，沙箱环境测试数据会进行定期数据清理。Beta 测试阶段每周日中午 12 点至每周一中午 12 点为维护时间，在此时间内沙箱环境部分功能可能不可用，敬请谅解。
- 请勿在沙箱进行压力测试，以免触发相应的限流措施，导致无法正常使用沙箱环境。
- 沙箱支持的各个开放产品，沙箱使用的特别说明请参见各产品的快速接入文档章节。

H1

如何使用沙箱环境

第一步：配置沙箱应用环境

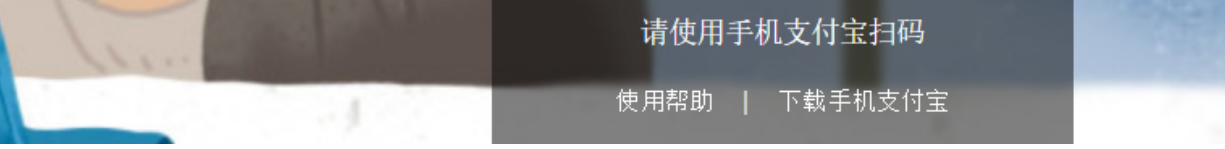
使用开发者账号登录 开放平台控制台 > **开发服务**， 点击 **研发服务** 即可进入 沙箱环境。

image.png

信息配置

必看部分

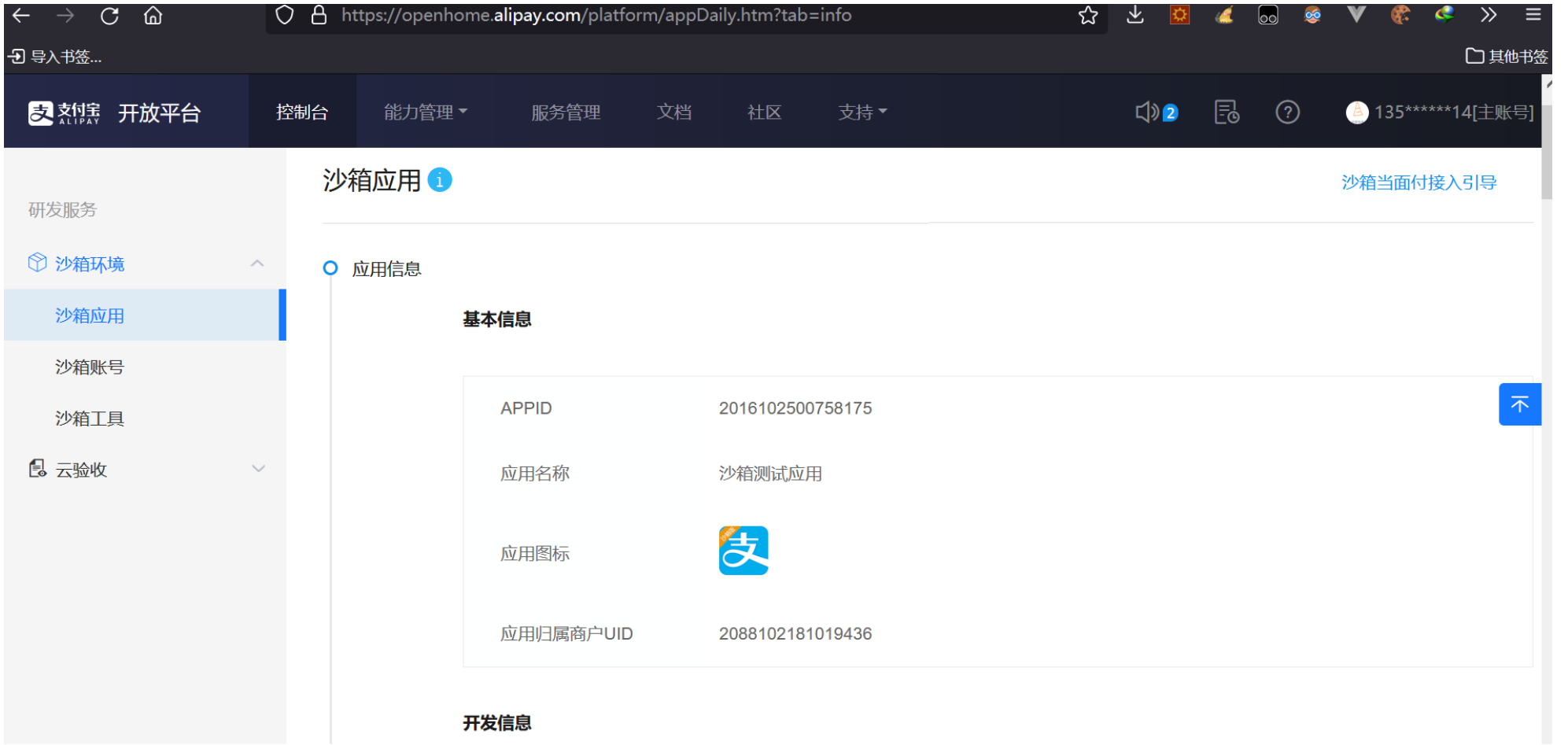
1. 进入沙箱环境页面，系统已经自动为你创建一个应用，在 **信息配置** 中可以看到应用信息。



7.跳转登陆页面，扫码登陆

8.跳转页面，点击研发服务

9.沙箱页面



- **沙箱应用 个人app以及公钥私钥信息**
- 沙箱账号 商家和买家信息
- 沙箱工具 手机沙箱支付宝app

10.**此时需要设置公钥和私钥，点击设置沙箱应用 查看密钥**

系统生成

- 应用公钥不用
- 应用私钥和支付宝公钥放程序

应用公钥：

复制公钥

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAr0dcj8Jx+iu54/ZxD5kPo9wBqc5BYnJd7CEs758WUTh0RQ
YzW350bWRrWOWZsc6nTmDtTwawvzToteKv0eGOfk1VMgEbtj8kzAA312U8XTDYXtp7ajvdhx7kGpPvLa/fvLGh+gvV
Of+rR9s3l5cU6Jna1itsP3JvXJwtnUlelktKg2Ww8ltyeasxwXY3oTUE2z4UwNoqy+5MhBc0fkJqGqBdCDEvlDg9E7/+AMz
Q8ZDl1TnBmfQlE719g6jyBGTDVBRZdNlmTl2dT7F6Kow3VS/ewBchWGnGxnGkXyGz6kxTSU15RnwIn8MTMrVGW
P4ggsdAsRhMNr+EC9U6/W9nmwIDAQAB

应用私钥：

JAVA语言 非JAVA语言

复制私钥

MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCvR1yPwnH6K7nj9nEPmQ+j3AGpzkFicl3slSzvnx
ZROHRFBjNbfnRtZGtY5ZmxzqdOYO1PBrC/NOi14q/R4Y5+TVUyARu2PyTMADfXZTxdMNhe2ntqO92HHuQak+8tr
9+8saH6C9U5/6tH2zeXlxTomdrWK2w/cm9cnC2dSV6WS0qDZbDyW3J5qzHBdjehNQTbPhTA2irL7kyEFzR+Qmoao
F0IMS+UOD0Tv/4AzNDxkOXVOcGZ9AgTvX2DqPlEZMNUFFI00iZOXZ1PsXoqjDdVL97AFyFYacbGcaRflbPqTFNJT
XlGfCWfwxMytUZY/iCCx0CxGEwPv4QL1Tr9b2ebAgMBAAECggEBAJ3V7fUKjOZRT8CG4ALCuwG8EMdfEyPahtcmf
Ld+k3MqGhE55y9hrOgBAGV5w7AlvkHnKGr8RFa6B0AaVGbz7QOxEaHlu7Kel72MCLzuza/BV2X66h2csPyHBbMIO
cDzWc8qSgjHdQgYQZ2gOuxHwS0/tr8bdQuhOcS0YqL7ObCtuGkiS191ssXMZ8H1JcCOGOQoANErCWBo3TnnQ7
WHCs26aubhiHC/P3HH5YG6sfkVtE6Q3CtQZTdfDdp5JKCDAGbhbccEGH7mSR2epOQqPPJudEt59AjUiVPAgR6LrKAK
QCXQMrgVAam+DAAQwUxyxcenJcS8SPlqj3Km/H4tOSewkCgYEA15A5UyhUEsmWqUsVjjbo8MDVdgLPJlUhWdUjgd
BtJ0V6RPOOVX4AqCsxK8uOL8dKyVafhS3cFXj9/A4Ln05kDmHzaq8HmpVzDh75v6qPCLpHlFYwaRKbuBV50Eadhj
oXzxfXrP2ytuSJ24qw8GxpcL3vu4j6md9UjB5C4VNZviUCgYEA0CiXW6SbCMJFwuEKyTjLQA7L75u+ej6HTynnFuTF/
GdefhE1E3wBv2yCqSoia+5+tUff8W/XBrqji/35vkybz12om6xSiPsA+2CWB2iewLKShrkKlCjD0B/tTkB00yk5Kp9Rrb9R
2+R9gnDlcK35Q/kpg6J9iojbDXCWV8IRQr8CgYAjOHllUO2LtExdlNsMxYQB8j/ubD8UcExmtSB8EWYESKHVqkXTSic
pUwVRtoyg2dsqUnD1/awBBezZxiG4AcLvZqmzAK4pX8nQwNblriR4tJcFcZ+vS08OlNQqZ59NUyiYubiPl1Qf9e9gY6TN
Y6lGtoH5DkSD562Ux0z+K/Q50QKBgAW3l7CkrL1o2JGprKoqLSrtPRyvomOpl/5AoGA4mFd+X1lHFvYaCdJAlLaUua
1NN52vDy+KatV8etF1gE6BvCXN5Luh2AJ7lmPFG1nlvdN9FBkvK+M122rz9YiYrJzAHRnWll+3SYBlnkne3+5gWOBN
D57V76zGZagA16BbEfxAoGAKB37M5bmqDAQD5CSFOl+mHosAc3U93yw8qRG/+5ahHE/buYP6lfkX5m6ikC5b06
skF+6a9O89Y4qgcurJsBRHd831zK5cb2kGGDeTbzz8tnXJ2R0mboAJANbPucJgdh+ePHfbinyTT4vVLNSxS5UJSfsX
BaUkLLCv6GbdrIzim8=

支付宝公钥：

复制公钥

11.需要的话可下载沙箱支付宝手机版，可扫码支付（账号密码查看沙箱账号）

集成项目

1.导入支付宝依赖

```
<dependency>
    <groupId>com.alipay.sdk</groupId>
    <artifactId>alipay-sdk-java</artifactId>
    <version>4.8.10.ALL</version>
</dependency>
```

2.封装配置类 AlipayConfig.java

```
public class AlipayConfig {
  /**
   * 服务网关    沙箱环境都是这个
     (去沙箱应用中复制该信息 沙箱应用/支付宝网关 )
   */
  public static String serverUrl = "https://openapi.alipaydev.com/gateway.do";

  /**
   * 应用id  后期可以替换成自己的id
    (去沙箱应用中复制该信息 沙箱应用/APPID)
   */
  public static String appId = "2016102500758175";

  /**
   RSA2密钥（公钥模式)
   应用私钥    后期替换成自己的私钥
   */
  public static String privateKey ="";
  /**
   *  发送数据的格式 目前只能为json
   */
  public static String format = "json";

  /**
   *  设置字符集编码 目前只能为utf-8
   */
  public static String charset = "utf-8";

  /**
      RSA2密钥（公钥模式)
     支付宝公钥 后期替换成自己的支付宝公钥
   */
  public static String publicKey = "";
  /**
   *  支付宝签名 目前是 RSA2
   */
  public static String signType = "RSA2";

  /**
   *  页面跳转同步通知页面路径 需http://格式的完整路径，不能加?id=123这类自定义参数，必须外网可以正常
访问
   *  付款结束后跳转的页面
   */
  public static String return_url = "http://localhost:8080/estore_alipay/index.jsp";

  public static AlipayClient getAlipayClient() {
      // 获得初始化的AlipayClient
      AlipayClient alipayClient = new DefaultAlipayClient
              (AlipayConfig.serverUrl, AlipayConfig.appId,
```

```
                AlipayConfig.privateKey, AlipayConfig.format,
                AlipayConfig.charset, AlipayConfig.publicKey, AlipayConfig.signType);
        return alipayClient;
    }
}
```

3.引入支付接口(注意配置访问路径)

```
public class PayServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //拿到订单id
         String orderid = request.getParameter("id");
        //调用service去数据库中查询当前订单
         OrderServiceImpl ordImpl = new OrderServiceImpl();
         OrderForm orderForm =
                 ordImpl.findOrderFormById(Integer.parseInt(orderid));

//支付
//浏览器动态产生一个付钱的二维码
 /**
  * 需要将我们的项目接入支付宝
  * 入驻蚂蚁金服平台
  */

 try {
      AlipayClient alipayClient =
              AlipayConfig.getAlipayClient();
      //设置请求参数
      AlipayTradePagePayRequest alipayRequest =
              new AlipayTradePagePayRequest();

      AlipayTradePayModel model =
              new AlipayTradePayModel();
      // 设定订单号 必须要写,且订单号不能重复，目前已经只是做测试，已经写死
      model.setOutTradeNo(System.currentTimeMillis()+"");
      // 设置订单金额
      model.setTotalAmount(orderForm.getCost()+"");
      // 订单名字
      model.setSubject("书籍订单");
      // 订单描述
      model.setBody(System.currentTimeMillis()+"");
```

```
                // 产品码
                model.setProductCode("FAST_INSTANT_TRADE_PAY");
                // 设置参数
                alipayRequest.setBizModel(model);
                // 设置回调地址
                alipayRequest.setReturnUrl(AlipayConfig.return_url);
                String result = alipayClient.pageExecute(alipayRequest).getBody();
                response.setContentType("text/html;charset=utf-8");
                response.getWriter().println(result);
            } catch (Exception e) {
                    e.printStackTrace();
            }
    }
}
```

4.此时需要添加业务 根据id查看订单信息

```
@Override
    public OrderForm findOrderFormById(int parseInt) {
        return orderFormMapper.findOrderFormById(parseInt);
    }
```

```
<select id="findOrderFormById" resultType="com.briup.bean.OrderForm">
         select *  from es_orderform
        where id = #{id}
    </select>
```

5.修改提交订单接口

先跳转到支付页面 支付完成回调订单列表

支付方案

- 下单再减库存(此处采用这种方式)
- 支付再减库存

SubmitOrderServlet

```
//resp.sendRedirect("/findAllOrder");
 resp.sendRedirect("/pay?id="+orderForm.getId());
```

AlipayConfig

```
/**
    *  页面跳转同步通知页面路径 需http://格式的完整路径，不能加?id=123这类自定义参数，必须外网可以正常访问
    *  付款结束后跳转的页面
    */
  public static String return_url = "http://localhost:8888/findAllOrder";
```

6.效果

书籍订单

收款方：沙箱环境

80.00 元

订单详情

红包
0 个
维护

账户余额 99780 元

支付 80.00 元

中国建设银行 信用卡 | 快捷 推荐

其他付款方式 添加快捷/网银付款

安全设置检测成功！无需短信校验。

支付宝支付密码：

忘记密码？

买家信息

买家账号 [redacted]

登录密码 [redacted]

支付密码 [redacted]

用户名称 [redacted]

证件类型 身份证(IDENTITY_CARD)

证件号码 027687192804052294

账户余额 99700.00 充值 取现

优化(todo)

自动登录

login.jsp

```
 <div class="fg_box">
                        <span style="line-height: 22px">
                            <input type="checkbox" name="autologin"
value="t">  七天内免登录
                        </span>
                        <a class="treg" href="#">立即注册</a>
                    </div>
```

LoginServlet

```
package com.briup.estore.web.servlet;

import com.briup.estore.constant.EstoreConstant;
import com.briup.estore.entity.Customer;
import com.briup.estore.service.IUserService;
import com.briup.estore.service.impl.UserServiceImpl;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/5/31-05-31-15:42
 * @Description: 登录接口
 */
@WebServlet("/userLogin")
public class UserLoginServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 1.获取用户名和密码
        String name = req.getParameter("name");
        String password = req.getParameter("password");
        String auto = req.getParameter("auto");
        // 2.调用service校验用户名和密码
        IUserService userService = new UserServiceImpl();
        // 3.跳转页面
        String path = "index.jsp";
        HttpSession session = req.getSession();
```

```
            try {
                // 登录成功  跳转首页(重定向)
                Customer customer = userService.login(name, password);
                session.setAttribute("customer", customer);
                // 如果勾选了自动登录 写入cookie
                if("true".equals(auto)){
                    Cookie nameCookie = new Cookie("auto_name",name);
                    nameCookie.setMaxAge(EstoreConstant.auto_max_time);
                    Cookie passwordCookie = new Cookie("auto_password",password);
                    passwordCookie.setMaxAge(EstoreConstant.auto_max_time);
                    resp.addCookie(nameCookie);
                    resp.addCookie(passwordCookie);
                }
            } catch (Exception e) {
                e.printStackTrace();
                // 登录失败  重新登录(请求转发 提示信息)
                path = "login.jsp";
                session.setAttribute("error", "登录失败 " +e.getMessage());
            }
            resp.sendRedirect(path);
        }
    }
```

AutoLogin

```
package com.briup.estore.web.filter;

import com.briup.estore.entity.Customer;
import com.briup.estore.service.IUserService;
import com.briup.estore.service.impl.UserServiceImpl;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/14-06-14-11:26
 * @Description: 自动登录
 */
@WebFilter("/*")
public class AutoLogin implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
```

```
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        // 勾选了自动登录后 会将用户写入到cookie中
        // 拦截器判断cookie中是否有cookie (cookie中默认有jsessionId)
        Cookie[] cookies = req.getCookies();
        if (cookies != null && cookies.length > 0) {
            String nameCookie = null;
            String passwordCookie = null;
            IUserService userService = new UserServiceImpl();
            for (Cookie cookie : cookies) {
                if ("auto_name".equals(cookie.getName())) {
                    nameCookie = cookie.getValue();
                }
                if ("auto_password".equals(cookie.getName())) {
                    passwordCookie = cookie.getValue();
                }
            }
            if (nameCookie != null && passwordCookie != null) {
                Customer customerFromDB = userService.login(nameCookie,
passwordCookie);
                req.getSession().setAttribute("customer", customerFromDB);
            }
        }
        // 如果有cookie.获取cookie中的值 并存入session中
        chain.doFilter(request, response);

    }

    @Override
    public void destroy() {

    }
}
```

事务拦截

MyBatisSqlSessionFactory

```
package com.briup.estore.utils;

import java.io.IOException;
import java.io.InputStream;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
```

```java
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class MyBatisSqlSessionFactory {
    private static ThreadLocal<SqlSession> threadLocal = new ThreadLocal<>();
    private static SqlSessionFactory sqlSessionFactory;

    public static SqlSessionFactory getSqlSessionFactory() {

        if(sqlSessionFactory==null) {
            InputStream inputStream = null;

            try {
                inputStream = Resources.getResourceAsStream("mybatis-config.xml");
                sqlSessionFactory = new
SqlSessionFactoryBuilder().build(inputStream);
            } catch (IOException e) {
                e.printStackTrace();
            }

        }

        return sqlSessionFactory;
    }


    public static SqlSession openSession() {
        SqlSession session = threadLocal.get();
        if (session == null) {
            session = openSession(false);
            threadLocal.set(session);
        }
        return session;
    }

    /**
     * 关闭连接
     */
    public static void closeSession(){
        SqlSession session = threadLocal.get();
        if (session != null) {
            session.close();
            threadLocal.remove();
        }
    }

    public static SqlSession openSession(boolean autoCommit) {
        return getSqlSessionFactory().openSession(autoCommit);
    }

}
```

TransactionFilter

```
package com.briup.estore.web.filter;

import com.briup.estore.utils.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import java.io.IOException;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/14-06-14-16:41
 * @Description: 事务过滤器(提交事务 回滚事务 关闭会话)
 */
@WebFilter("/*")
public class TransactionFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
        // 手动提交
        SqlSession session = MyBatisSqlSessionFactory.openSession();
        try {
            // 执行对应代码
            chain.doFilter(request,response);
            // 提交事务
            session.commit();
        } catch (Exception e) {
            // 事务归滚
            session.rollback();
            e.printStackTrace();
        }finally {
            // 关闭session
            MyBatisSqlSessionFactory.closeSession();
        }


    }

    @Override
    public void destroy() {

    }
```

```
}
```

使用

```
package com.briup.estore.web.servlet;

import com.briup.estore.entity.Customer;
import com.briup.estore.service.IUserService;
import com.briup.estore.service.impl.UserServiceImpl;
import org.apache.taglibs.standard.tag.common.sql.DataSourceUtil;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/5/31-05-31-10:33
 * @Description:  用户注册接口 (接收/响应数据)
 */
@WebServlet("/userRegister")
public class UserRegisterServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 1.接收参数 并封装Customer对象
        String name = req.getParameter("name");
        String password = req.getParameter("password");
        String zipCode = req.getParameter("zipCode");
        String address = req.getParameter("address");
        String telephone = req.getParameter("telephone");
        String email = req.getParameter("email");
        Customer customer = new
Customer(name,password,zipCode,address,telephone,email);
        // 2.交给service判断 用户名不能为空
        IUserService userService = new UserServiceImpl();
        // 3.判断是否注册成功
        String path="login.jsp";
        try {
            userService.register(customer);
        } catch (Exception e) {
            e.printStackTrace(); // 打印异常信息
            // 失败 重新注册  请求转发
            // 将该错误信息存到request中,将来页面取出
            req.getSession().setAttribute("error","注册失败 " +e.getMessage());
            path = "register.jsp";
            // 事务可以向上抛 并回滚,但是页面不正常跳转了
```

```
            throw new RuntimeException(e);
        }finally {
            // 确保是否发生异常 都会正常执行
            resp.sendRedirect(path);
        }
    }
}
```

注意 try catch finally块,确保事务和页面跳转能够正常运转

service层模拟异常

```
package com.briup.estore.service.impl;

import com.briup.estore.dao.ICustomerDao;
import com.briup.estore.entity.Customer;
import com.briup.estore.service.IUserService;
import com.briup.estore.utils.MD5Util;
import com.briup.estore.utils.MyBatisSqlSessionFactory;
import org.apache.ibatis.session.SqlSession;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/5/31-05-31-11:01
 * @Description: 顾客业务实现
 */
public class UserServiceImpl implements IUserService {
    SqlSession session = MyBatisSqlSessionFactory.openSession();
    ICustomerDao customerDao = session.getMapper(ICustomerDao.class);

    @Override
    public void register(Customer customer) {
        String name = customer.getName();
        // 1.校验用户
            // 1.用户名不能为空
        if (name == null || "".equals(name.trim())) {
            // 参数是message new RuntimeException(message);
            throw new RuntimeException("用户名不能为空");
        }
            // 2.用户名不能已经在数据库存在 (携带用户名去数据库查)
        Customer cusomerFromDB = findByName(name);
        if (cusomerFromDB != null) {
            throw new RuntimeException("用户名已经存在");
        }
        // 2.注册用户(调用dao)
        // 加密
        customer.setPassword(MD5Util.encode(customer.getPassword()));
        customerDao.register(customer);
//        int i = 1/0;
```

```
        // 注意事务的提交
        //session.commit();
    }

    /**
     *
     * @param name 用户名称
     * @return 该用户对象
     */
    public Customer findByName(String name){
        // 去数据库查
        return customerDao.findByName(name);
    }

    /**
     * 登录功能
     * @param name 用户名
     * @param password 密码
     * @return 用户对象    select * from es_user where username=#{} and password=#{}
     */
    @Override
    public Customer login(String name, String password) {
        // 用户名不对(带着用户名去找 是否有, 没有则用户名不对)
        Customer customerFromDB = findByName(name);
        if (customerFromDB == null) {
            throw new RuntimeException("用户名有误");
        }
        // 密码不对(比较页面传的密码和查出来的对象的密码)
        if(!MD5Util.encode(password).equals(customerFromDB.getPassword())){
            throw new RuntimeException("密码有误");
        }
        return customerFromDB;
    }
}
```

导出数据

orderList.jsp

```
<!--头部导航-->
<div class="nav_top">
    <div class="nav_top_center">
        <div>
            订单列表
        </div>
        <div>
            <%--<input type="button" value="导出" onclick="exportData()">--%>
            <span style="font-size:14px;font-weight: bold;color: #FFF"
onclick="exportData()">导出</span>
        </div>
    </div>
</div>
```

OrderListExport

```
package com.briup.estore.web.servlet;

import cn.hutool.core.collection.CollUtil;
import cn.hutool.poi.excel.ExcelUtil;
import cn.hutool.poi.excel.ExcelWriter;
import com.briup.estore.dto.ExportData;
import com.briup.estore.entity.Customer;
import com.briup.estore.entity.OrderForm;
import com.briup.estore.service.IOrderService;
import com.briup.estore.service.impl.OrderServieImpl;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * @Auther: vanse(lc))
 * @Date: 2022/6/14-06-14-9:20
 * @Description: 导出excel表格
 */
@WebServlet("/exportData")
public class OrderListExport extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
```

```
        Customer customer = (Customer) req.getSession().getAttribute("customer");
        IOrderService orderService = new OrderServieImpl();
        List<OrderForm> orderFormList =
orderService.findOrderListByCid(customer.getId());
        List<ExportData> exportDataList = new ArrayList<>();
        orderFormList.forEach(orderForm -> {
            ExportData exportData = new ExportData();
            exportData.setOrderId(orderForm.getId());
            exportData.setCost(orderForm.getCost());
            exportData.setOrderDate(orderForm.getOrderdate());
            exportData.setReceiveName(orderForm.getShopAddress().getReceiveName());
            exportDataList.add(exportData);
        });

        try {
            ArrayList<ExportData> rows = CollUtil.newArrayList(exportDataList);
            // 通过工具类创建writer
            long time = System.currentTimeMillis();
            ExcelWriter writer = ExcelUtil.getWriter("d:/order-"+time+".xlsx");
//          ExcelWriter writer = ExcelUtil.getWriter();
            // 合并单元格后的标题行，使用默认标题样式
            writer.merge(3, "账单");
            // 一次性写出内容，使用默认样式，强制输出标题
            writer.write(rows, true);
            // 关闭writer，释放内存
            writer.close();

            resp.setContentType("text/plain;charset=utf-8");
            resp.getWriter().write("导出成功");
        } catch (IOException e) {
            e.printStackTrace();
            resp.getWriter().write("导出失败");
        }
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        doGet(req, resp);
    }
}
```

抽取servlet

...