# 1.通用流程

## 1.1项目准备（SSM+IDEA）

### 1.1.1pom

|  |
| --- |
| <packaging>war</packaging>  <properties>  <maven.compiler.source>1.8</maven.compiler.source>  <maven.compiler.target>1.8</maven.compiler.target>  <spring.version>4.3.13.RELEASE</spring.version>  <mybatis.version>3.4.5</mybatis.version>  <!-- log4j日志文件管理包版本 -->  <slf4j.version>1.7.21</slf4j.version>  <log4j.version>1.2.17</log4j.version>  </properties>  <dependencies>  <!-- 日志处理 -->  <!-- log start -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>${slf4j.version}</version>  </dependency>  <!-- log end -->  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aop</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>${spring.version}</version>  </dependency>  <!-- spring 整合 mybatis包 -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.3.2</version>  </dependency>  <!-- spring 测试-->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>${spring.version}</version>  </dependency>  <!-- mybatis -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>${mybatis.version}</version>  </dependency>  <!-- MySql -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.40</version>  </dependency>  <!-- 连接池 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.0.28</version>  </dependency>  <!-- JSP、servlet相关 -->  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.1.0</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jsp-api</artifactId>  <version>2.0</version>  <scope>provided</scope>  </dependency>  <!-- Jackson Json处理工具包 -->  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.8.10</version>  </dependency>  <!--JSONObject工具包-->  <dependency>  <groupId>net.sf.json-lib</groupId>  <artifactId>json-lib</artifactId>  <version>2.4</version>  <classifier>jdk15</classifier>  </dependency>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-lang3</artifactId>  <version>3.7</version>  </dependency>  <dependency>  <groupId>commons-beanutils</groupId>  <artifactId>commons-beanutils</artifactId>  <version>1.8.3</version>  </dependency>  <dependency>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  <version>1.1.1</version>  </dependency>  <dependency>  <groupId>commons-collections</groupId>  <artifactId>commons-collections</artifactId>  <version>3.2.1</version>  </dependency>  <dependency>  <groupId>net.sf.ezmorph</groupId>  <artifactId>ezmorph</artifactId>  <version>1.0.6</version>  </dependency>  <!-- 单元测试 -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  <scope>test</scope>  </dependency>  </dependencies>  <build>  <!-- 拷贝资源文件 -->  <!-- 添加此节点将mybatis的XxxMapper.xml文件发布到target路径下 -->  <resources>  <resource>  <directory>src/main/java</directory>  <includes>  <!-- <include>\*\*/\*.properties</include> -->  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  <!-- 上面的配置导致src/main/resources的所有文件都不能被扫描，因此还要配多一个 -->  <resource>  <directory>src/main/resources</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  </resources>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <configuration>  <forkMode>once</forkMode>  <argLine>-Dfile.encoding=UTF-8</argLine>  </configuration>  </plugin>  </plugins>  </build>  </project> |

### 1.1.2 web.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  version="3.1">  <!-- spring：配置spring容器监听器 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/spring-\*.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 配置SpringMVC的 DispatcherServlet-->  <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- 配置 DispatcherServlet 的一个初始化参数: 配置 springmvc 配置文件的位置和名称 -->  <!--  实际上也可以不通过 contextConfigLocation 来配置 springmvc 的配置文件, 而使用默认的.  默认的配置文件为: /WEB-INF/<servlet-name>-servlet.xml  如：springmvc-servlet.xml  -->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/springmvc-servlet.xml</param-value>  <!-- <param-value>/WEB-INF/springmvc-servlet.xml</param-value> -->  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <!-- 所有的客户端请求由 DispatcherServlet处理：前端控制器模式-->  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  <!-- 字符编码过滤器 : 放在其他过滤器的前面-->  <filter>  <filter-name>CharacterEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!--jfreeChart生成图表所用-->  <!--  <servlet>  <servlet-name>DisplayChart</servlet-name>  <servlet-class>org.jfree.chart.servlet.DisplayChart</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>DisplayChart</servlet-name>  <url-pattern>/chart</url-pattern>  </servlet-mapping>  -->  <!--  配置 org.springframework.web.filter.HiddenHttpMethodFilter: 可以把 POST 请求转为 DELETE 或 PUT 请求  -->  <!--<filter>-->  <!--<filter-name>HiddenHttpMethodFilter</filter-name>-->  <!--<filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>-->  <!--</filter>-->  <!--<filter-mapping>-->  <!--<filter-name>HiddenHttpMethodFilter</filter-name>-->  <!--<url-pattern>/\*</url-pattern>-->  <!--</filter-mapping>-->  </web-app> |

### 1.1.3 log4j.propertries

|  |
| --- |
| log4j.rootLogger=INFO, stdout  log4j.appender.stdout=org.apache.log4j.ConsoleAppender  log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %t %c{2}:%L - %m%n  # Spring  #log4j.category.org.springframework.beans.factory=INFO  log4j.logger.org.springframework=INFO  #mybatis  log4j.logger.org.mybatis = ERROR  #打印sql语句  log4j.logger.api.dao = DEBUG  #hibernate  log4j.logger.org.hibernate=INFO  # General Apache libraries  log4j.logger.org.apache=INFO  # Default Shiro logging  log4j.logger.org.apache.shiro=TRACE  # Disable verbose logging  log4j.logger.org.apache.shiro.util.ThreadContext=INFO  log4j.logger.org.apache.shiro.cache.ehcache.EhCache=INFO |

### 1.1.4 db.properties

|  |
| --- |
| jdbc.jdbcUrl=jdbc:mysql://localhost:3306/study\_practice?useUnicode=true&characterEncoding=utf8&allowMultiQueries=true  jdbc.driverClass=com.mysql.jdbc.Driver  jdbc.user=root  jdbc.password=root |

### 1.1.5springmvc-servlet.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context" xmlns:p="http://www.springframework.org/schema/p"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd">  <!-- 扫描包下所有标注@Controller的组件 -->  <context:component-scan base-package="com.custom.controller"/>  <!--扫描切面-->  <!--<context:component-scan base-package="api.exception"/>-->  <!-- 视图解析器 -->  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"  p:prefix="/WEB-INF/jsp/"  p:suffix=".jsp" />  <!-- 配置注解驱动 -->  <mvc:annotation-driven />  <!-- 处理静态资源 -->  <mvc:default-servlet-handler/>  </beans> |

### 1.1.6spring-service.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx"  xmlns:aop="http://www.springframework.org/schema/aop"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">  <!-- 扫描service -->  <context:component-scan base-package="com.custom.service" />  <!--事务管理器-->  <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  <property name="dataSource" ref="dataSource" />  </bean>  <!-- 通知 -->  <tx:advice id="txAdvice" transaction-manager="transactionManager">  <tx:attributes>  <!-- 传播行为 -->  <!-- REQUIRED：如果有事务，则在事务中执行；如果没有事务，则开启一个新的事物 -->  <tx:method name="save\*" propagation="REQUIRED" />  <tx:method name="insert\*" propagation="REQUIRED" />  <tx:method name="add\*" propagation="REQUIRED" />  <tx:method name="create\*" propagation="REQUIRED" />  <tx:method name="delete\*" propagation="REQUIRED" />  <tx:method name="update\*" propagation="REQUIRED" />  <!-- SUPPORTS：如果有事务，则在事务中执行；如果没有事务，则不会开启事物 -->  <tx:method name="find\*" propagation="SUPPORTS" read-only="true" />  <tx:method name="select\*" propagation="SUPPORTS" read-only="true" />  <tx:method name="get\*" propagation="SUPPORTS" read-only="true" />  <tx:method name="query\*" propagation="SUPPORTS" read-only="true" />  <tx:method name="\*" propagation="SUPPORTS" read-only="true" />  </tx:attributes>  </tx:advice>  <!-- 切面 -->  <aop:config>  <!-- 定义在service包或者子包里的任意方法的执行 -->  <aop:pointcut id="txPointCut"  expression="execution(\* api.service..\*.\*(..))" />  <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointCut" />  </aop:config>  </beans> |

### 1.1.7spring-dao.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:util="http://www.springframework.org/schema/util"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util.xsd">  <!-- 扫描dao -->  <context:component-scan base-package="com.custom.dao" />  <!-- 加载数据库配置文件和其他配置文件 -->  <context:property-placeholder location="classpath:db.properties" />  <!-- 数据库连接池 -->  <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"  destroy-method="close">  <property name="url" value="${jdbc.jdbcUrl}" />  <property name="username" value="${jdbc.user}" />  <property name="password" value="${jdbc.password}" />  <property name="driverClassName" value="${jdbc.driverClass}" />  <!-- 初始化连接大小 -->  <property name="initialSize" value="0" />  <!-- 连接池最大使用连接数量 -->  <property name="maxActive" value="20" />  <!-- 连接池最小空闲 -->  <property name="minIdle" value="0" />  <!-- 获取连接最大等待时间 -->  <property name="maxWait" value="60000" />  </bean>  <!--整合mybatis：sqlSessionFactory-->  <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  <property name="dataSource" ref="dataSource" />  <property name="configLocation" value="classpath:mybatis/mybatis-config.xml" />  </bean>  <!--配置dao层：扫描Mapper文件，用于生成Mapper的代理对象-->  <!--<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">-->  <!--&lt;!&ndash;<property name="basePackage" value="api.dao" />&ndash;&gt;-->  <!--<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />-->  <!--</bean>-->  </beans> |

### 1.1.8 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>  <!--<typeAliases>-->  <!--&lt;!&ndash;方法一：一次定义一个对象别名，每个对象需要单独设置&ndash;&gt;-->  <!--<typeAlias type="pojo.TbUser" alias="TbUser"/>-->  <!--&lt;!&ndash;方法二：设置包扫描&ndash;&gt;-->  <!--<package name="pojo"/>-->  <!--</typeAliases>-->  <environments default="development">  <environment id="development">  <transactionManager type="JDBC"/>  <dataSource type="POOLED">  <property name="driver" value="com.mysql.jdbc.Driver"/>  <property name="url" value="jdbc:mysql://localhost:3306/study\_practice?characterEncoding=UTF-8"/>  <property name="username" value="root"/>  <property name="password" value="123"/>  </dataSource>  </environment>  </environments>  <!--<mappers>-->  <!--<package name="api.dao"/>-->  <!--</mappers>-->  </configuration> |

# 2.FreeChart

学习网址：

<https://blog.csdn.net/qq_28334041/article/details/70308071#%E5%9B%BE%E7%89%87%E6%B8%B2%E6%9F%93%E5%AF%B9%E8%B1%A1>

## 2.1 FreeChart小Demo

### 2.1.1项目准备

根据1.1内容进行项目准备

### 2.1.2小Demo实现的重要节点

#### 2.1.2.1实现前提

使用FreeChart需要先引入两个pom

|  |
| --- |
| <dependency>  <groupId>org.jfree</groupId>  <artifactId>jfreechart</artifactId>  <version>1.0.19</version>  </dependency>  <dependency>  <groupId>org.jfree</groupId>  <artifactId>jcommon</artifactId>  <version>1.0.23</version>  </dependency> |

有的网友也提示要引入gnujaxp的pom，但是我引入后，程序报错，所以我把这个包给注释掉了。

#### 2.1.2.2创建集合与制图对象

编写工具类CharUtil，ChartUtil类是一个自定义的制图工具类，主要作用是提供freechart作图的两个重要元素：图形的数据集合和创建图表的制图对象。

其中createDataSet()方法用于创建柱形图所需要的数据集合，它返回CategoryDataset对象。CreateChart()用于创建制图对象JFreeChart，在此方法中通过ChartFactory对象的createBarChart3D()方法创建一个3D效果的柱形图对象，并将其返回。

|  |
| --- |
| package com.custom.utils;  import org.jfree.chart.ChartFactory;  import org.jfree.chart.JFreeChart;  import org.jfree.chart.StandardChartTheme;  import org.jfree.chart.plot.PlotOrientation;  import org.jfree.data.category.CategoryDataset;  import org.jfree.data.category.DefaultCategoryDataset;  import java.awt.\*;  /\*\*  \* Created by user on 2018/9/11.  \* ChartUtil类是一个自定义的制图工具类,  \* 其中createDataSet()方法用于创建柱形图所需要的数据集合，它返回CategoryDataset对象。  \* CreateChart()用于创建制图对象JFreeChart，在此方法中通过ChartFactory对象的createBarChart3D()方法创建一个3D效果的柱形图对象，并将其返回。  \*/  public class CharUtil {  //创建数据集合  public static CategoryDataset createDataSet(){  DefaultCategoryDataset dataSet=new DefaultCategoryDataset(); //实例化DefaultCategoryDataset对象  dataSet.addValue(500, "JAVA图书", "J2SE类"); //向数据集合中添加数据  dataSet.addValue(100, "JAVA图书", "JAVAME类");  dataSet.addValue(900, "JAVA图书", "J2EE类");  return dataSet;  }  //创建JFreeChart 对象  public static JFreeChart createChart (){  StandardChartTheme standardChartTheme=new StandardChartTheme("CN"); //创建主题样式  standardChartTheme.setExtraLargeFont(new Font("隶书",Font.BOLD,20)); //设置标题字体  standardChartTheme.setRegularFont(new Font("宋体",Font.PLAIN,15)); //设置图例的字体  standardChartTheme.setLargeFont(new Font("宋体", Font.PLAIN, 15)); //设置轴向的字体  ChartFactory.setChartTheme(standardChartTheme); //设置主题样式  //通过ChartFactory 创建JFreeChart  JFreeChart chart = ChartFactory.createBarChart3D(  "JAVA图书销量统计", //图表标题  "JAVA图书", //横轴标题  "销量（本）", //纵轴标题  createDataSet(), //数据集合  PlotOrientation.VERTICAL, //图表方向  false, //是否显示图例标识  false, //是否显示tooltips  false); //是否支持超链接  return chart;  }  } |

#### 2.1.2.3创建管理图片的servlet

JFreeChart组件能够生成.jpeg、.png格式的图片，其输出方式可以直接存储在硬盘之中，也可以交给JFreeChart组件进行管理。在Web项目的应用中，所生成的图形报表一般均为动态图表，如果对每次生成的图表都进行直接存储，将会产生大量的垃圾文件，时间长久将要做一定清理工作。因此，由JFreeChart组件管理所生成的图片是一个不错的选择，它不仅可以提供图片的路径，而且在查看图片后，JFreeChart组件自动做出清理工作。

JFreeChart组件提供了一个Servlet文件用于获取生成的图片，此Servlet文件存在于JFreeChart组件包中，所以在使用过程中，需要将其配置到Web.xml文件中，其配置方法如下：

|  |
| --- |
| <!--jfreeChart生成图表所用-->  <servlet>  <servlet-name>DisplayChart</servlet-name>  <servlet-class>org.jfree.chart.servlet.DisplayChart</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>DisplayChart</servlet-name>  <url-pattern>/chart</url-pattern>  </servlet-mapping> |

#### 2.1.2.4创建jsp处理显示图片

ServletUtilities类的saveChartAsJPEG()方法进行生成，它返回一个.jpeg格式的图片名称。通过此方法生成图片后，调用已注册的JFreeChart提供的Servlet类DisplayChart，即可获取到图片的相对路径。

在使用saveChartAsJPEG()方法时，需要获得制图对象，所以还需要用到上面写的chartUtil工具类。

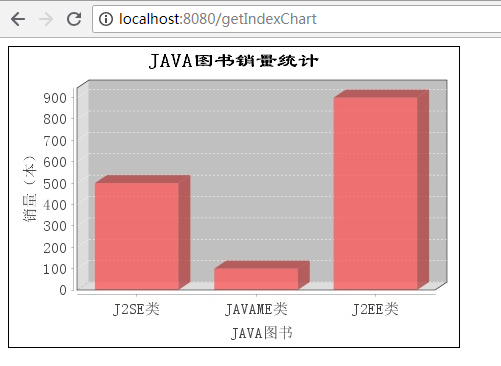
所以通过import标签直接在jsp中引入以上两个类。

|  |
| --- |
| <%--<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">--%>  <%@ page import="org.jfree.chart.servlet.ServletUtilities,com.custom.utils.CharUtil" %>  <%--  Created by IntelliJ IDEA.  User: user  Date: 2018/9/12  Time: 16:40  To change this template use File | Settings | File Templates.  --%>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Java图书销量统计</title>  </head>  <body>  <%--ServletUtilities类的saveChartAsJPEG()方法进行生成，它返回一个.jpeg格式的图片名称  参数的说明:  chart :制图对象JFreeChartford  width:所生成图片的宽度  height:所生成图片的高度  session”HttpSession对象  通过此方法生成图片后，调用已注册的JFreeChart提供的Servlet类DisplayChart，即可获取到图片的相对路径。  --%>  <%  String fileName = ServletUtilities.saveChartAsJPEG(CharUtil.createChart(),450,300,session);  String graphURL = request.getContextPath() + "/chart?filename=" + fileName;  %>  <img src="<%=graphURL%>" border="1">  </body>  </html> |

#### 2.1.2.5创建controller

|  |
| --- |
| package com.custom.controller;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  /\*\*  \* Created by user on 2018/9/12.  \*/  @Controller  public class JFreeChartController {  @RequestMapping("/getIndexChart")  public String getIndexChart(){  return "index";  }  } |

结果如下：



#### 2.1.2.6优化图片

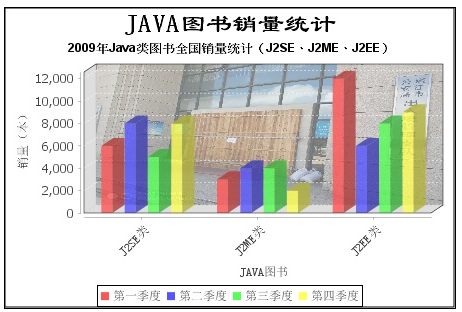
通过修改制图对象（charUtil工具类中）就可以实现显示的优化。

通常情况下，为了使所生成图片美观、大方，需要对所生成的图片进行一定的设置，如制图对象设置、绘图区设置、坐标轴设置及图片渲染等。

程序中涉及到了背景图片的设置，本程序是将图片放在了resources文件夹下的image文件夹中了，同时修改一下pom中的资源插件，以便于框架能够扫描到图片。

|  |
| --- |
| package com.custom.utils;  import org.jfree.chart.ChartFactory;  import org.jfree.chart.JFreeChart;  import org.jfree.chart.axis.CategoryAxis;  import org.jfree.chart.axis.CategoryLabelPositions;  import org.jfree.chart.axis.ValueAxis;  import org.jfree.chart.plot.CategoryPlot;  import org.jfree.chart.plot.PlotOrientation;  import org.jfree.chart.renderer.category.BarRenderer3D;  import org.jfree.chart.title.TextTitle;  import org.jfree.data.category.CategoryDataset;  import org.jfree.data.category.DefaultCategoryDataset;  import org.jfree.ui.VerticalAlignment;  import javax.imageio.ImageIO;  import java.awt.\*;  import java.io.IOException;  /\*\*  \* Created by user on 2018/9/11.  \* ChartUtil类是一个自定义的制图工具类,  \* 其中createDataSet()方法用于创建柱形图所需要的数据集合，它返回CategoryDataset对象。  \* CreateChart()用于创建制图对象JFreeChart，在此方法中通过ChartFactory对象的createBarChart3D()方法创建一个3D效果的柱形图对象，并将其返回。  \*/  public class CharUtil {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*简单的柱状图小Demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // //创建数据集合  // public static CategoryDataset createDataSet(){  // DefaultCategoryDataset dataSet=new DefaultCategoryDataset(); //实例化DefaultCategoryDataset对象  // dataSet.addValue(500, "JAVA图书", "J2SE类"); //向数据集合中添加数据  // dataSet.addValue(100, "JAVA图书", "JAVAME类");  // dataSet.addValue(900, "JAVA图书", "J2EE类");  // return dataSet;  // }  // //创建JFreeChart 对象  // public static JFreeChart createChart (){  // StandardChartTheme standardChartTheme=new StandardChartTheme("CN"); //创建主题样式  // standardChartTheme.setExtraLargeFont(new Font("隶书",Font.BOLD,20)); //设置标题字体  // standardChartTheme.setRegularFont(new Font("宋体",Font.PLAIN,15)); //设置图例的字体  // standardChartTheme.setLargeFont(new Font("宋体", Font.PLAIN, 15)); //设置轴向的字体  // ChartFactory.setChartTheme(standardChartTheme); //设置主题样式  // //通过ChartFactory 创建JFreeChart  // JFreeChart chart = ChartFactory.createBarChart3D(  // "JAVA图书销量统计", //图表标题  // "JAVA图书", //横轴标题  // "销量（本）", //纵轴标题  // createDataSet(), //数据集合  // PlotOrientation.VERTICAL, //图表方向  // false, //是否显示图例标识  // false, //是否显示tooltips  // false); //是否支持超链接  // return chart;  // }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*通过图表相关属性设置优化后的柱状图\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //创建数据集合  public static CategoryDataset createDataSet() {  //实例化DefaultCategoryDataset对象  DefaultCategoryDataset dataSet = new DefaultCategoryDataset();  //添加第一季度数据  dataSet.addValue(6000, "第一季度", "J2SE类");  dataSet.addValue(3000, "第一季度", "J2ME类");  dataSet.addValue(12000, "第一季度", "J2EE类");  //添加第二季度数据  dataSet.addValue(8000, "第二季度", "J2SE类");  dataSet.addValue(4000, "第二季度", "J2ME类");  dataSet.addValue(6000, "第二季度", "J2EE类");  //添加第三季度数据  dataSet.addValue(5000, "第三季度", "J2SE类");  dataSet.addValue(4000, "第三季度", "J2ME类");  dataSet.addValue(8000, "第三季度", "J2EE类");  //添加第四季度数据  dataSet.addValue(8000, "第四季度", "J2SE类");  dataSet.addValue(2000, "第四季度", "J2ME类");  dataSet.addValue(9000, "第四季度", "J2EE类");  return dataSet;  }  //创建JFreeChart 对象  public static JFreeChart createChart() {  //通过ChartFactory创建JFreeChart  JFreeChart chart = ChartFactory.createBarChart3D(  "JAVA图书销量统计", //图表标题  "JAVA图书", //横轴标题  "销量（本）", //纵轴标题  createDataSet(), //数据集合  PlotOrientation.VERTICAL, //图表方向  true, //是否显示图例标识  false, //是否显示tooltips  false); //是否支持超链接  //背景图片  Image image = null;  //File file = new File("/image/test.jpg");  try {  //创建背景图片  //image = ImageIO.read(file);  image = ImageIO.read(CharUtil.class.getResource("/image/test.jpg"));  } catch (IOException e) {  e.printStackTrace();  }  //设置标题字体  chart.getTitle().setFont(new Font("隶书",Font.BOLD,25));  //设置图例类别字体  chart.getLegend().setItemFont(new Font("宋体",Font.PLAIN,12));  chart.setBorderVisible(true); //设置显示边框  //实例化TextTitle对象  TextTitle subTitle = new TextTitle("2009年Java类图书全国销量统计（J2SE、J2ME、J2EE）");  //设置居中显示  subTitle.setVerticalAlignment(VerticalAlignment.BOTTOM);  chart.addSubtitle(subTitle); //添加子标题  //获取绘图区对象  CategoryPlot plot = chart.getCategoryPlot();  plot.setForegroundAlpha(0.8F); //设置绘图区前景色透明度  plot.setBackgroundAlpha(0.5F); //设置绘图区背景色透明度  plot.setBackgroundImage(image); //设置绘图区背景图片  //获取坐标轴对象  CategoryAxis categoryAxis = plot.getDomainAxis();  //设置坐标轴标题字体  categoryAxis.setLabelFont(new Font("宋体",Font.PLAIN,12));  //设置坐标轴标尺值字体  categoryAxis.setTickLabelFont(new Font("宋体",Font.PLAIN,12));  //设置坐标轴标题旋转角度  categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.UP\_45);  //获取数据轴对象  ValueAxis valueAxis = plot.getRangeAxis();  //设置数据轴字体  valueAxis.setLabelFont(new Font("宋体",Font.PLAIN,12));  //获取图片渲染对象  BarRenderer3D renderer = new BarRenderer3D();  renderer.setItemMargin(0.32); //设置柱子间的间距  plot.setRenderer(renderer); //设置图片渲染对象  return chart;  }  } |

测试结果如下：



#### 2.1.2.7 FreeChart整合JDBC

在实际开发过程中，数据集合中的数据大部分来自于数据库，所以在制图前，需要进行数据库操作来获取数据。JFreeChart组件对这一过程进行了封装，通过自定义的SQL语句就可获取到已封装好的数据集合对象。

常用的JDBC数据集合对象有三种，分别为：JDBCCategoryDataSet（JDBC填充类别数据集合）、JDBCPieDataSet（JDBC填充饼形图数据集合）、JDBCXYDataset（JDBC填充坐标轴数据集合）。它们的使用方法都非常简单。程序中会讲解使用方法。

我们尝试查询数据库来初始化数据集合，因此依然是在chartUtil中操作。

|  |
| --- |
| package com.custom.utils;  import org.jfree.chart.ChartFactory;  import org.jfree.chart.JFreeChart;  import org.jfree.chart.axis.CategoryAxis;  import org.jfree.chart.axis.CategoryLabelPositions;  import org.jfree.chart.axis.ValueAxis;  import org.jfree.chart.labels.StandardPieSectionLabelGenerator;  import org.jfree.chart.plot.CategoryPlot;  import org.jfree.chart.plot.PiePlot;  import org.jfree.chart.plot.PlotOrientation;  import org.jfree.chart.renderer.category.BarRenderer3D;  import org.jfree.chart.title.TextTitle;  import org.jfree.data.category.CategoryDataset;  import org.jfree.data.category.DefaultCategoryDataset;  import org.jfree.data.general.PieDataset;  import org.jfree.data.jdbc.JDBCPieDataset;  import org.jfree.ui.VerticalAlignment;  import javax.imageio.ImageIO;  import java.awt.\*;  import java.io.IOException;  import java.text.NumberFormat;  import java.util.Properties;  /\*\*  \* Created by user on 2018/9/11.  \* ChartUtil类是一个自定义的制图工具类,  \* 其中createDataSet()方法用于创建柱形图所需要的数据集合，它返回CategoryDataset对象。  \* CreateChart()用于创建制图对象JFreeChart，在此方法中通过ChartFactory对象的createBarChart3D()方法创建一个3D效果的柱形图对象，并将其返回。  \* 该类中还涉及到了整合数据库，通过查询数据库来初始化数据集合  \*/  public class CharUtil {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*简单的柱状图小Demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // //创建数据集合  // public static CategoryDataset createDataSet(){  // DefaultCategoryDataset dataSet=new DefaultCategoryDataset(); //实例化DefaultCategoryDataset对象  // dataSet.addValue(500, "JAVA图书", "J2SE类"); //向数据集合中添加数据  // dataSet.addValue(100, "JAVA图书", "JAVAME类");  // dataSet.addValue(900, "JAVA图书", "J2EE类");  // return dataSet;  // }  // //创建JFreeChart 对象  // public static JFreeChart createChart (){  // StandardChartTheme standardChartTheme=new StandardChartTheme("CN"); //创建主题样式  // standardChartTheme.setExtraLargeFont(new Font("隶书",Font.BOLD,20)); //设置标题字体  // standardChartTheme.setRegularFont(new Font("宋体",Font.PLAIN,15)); //设置图例的字体  // standardChartTheme.setLargeFont(new Font("宋体", Font.PLAIN, 15)); //设置轴向的字体  // ChartFactory.setChartTheme(standardChartTheme); //设置主题样式  // //通过ChartFactory 创建JFreeChart  // JFreeChart chart = ChartFactory.createBarChart3D(  // "JAVA图书销量统计", //图表标题  // "JAVA图书", //横轴标题  // "销量（本）", //纵轴标题  // createDataSet(), //数据集合  // PlotOrientation.VERTICAL, //图表方向  // false, //是否显示图例标识  // false, //是否显示tooltips  // false); //是否支持超链接  // return chart;  // }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*通过图表相关属性设置优化后的柱状图\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // //创建数据集合  // public static CategoryDataset createDataSet() {  // //实例化DefaultCategoryDataset对象  // DefaultCategoryDataset dataSet = new DefaultCategoryDataset();  // //添加第一季度数据  // dataSet.addValue(6000, "第一季度", "J2SE类");  // dataSet.addValue(3000, "第一季度", "J2ME类");  // dataSet.addValue(12000, "第一季度", "J2EE类");  // //添加第二季度数据  // dataSet.addValue(8000, "第二季度", "J2SE类");  // dataSet.addValue(4000, "第二季度", "J2ME类");  // dataSet.addValue(6000, "第二季度", "J2EE类");  // //添加第三季度数据  // dataSet.addValue(5000, "第三季度", "J2SE类");  // dataSet.addValue(4000, "第三季度", "J2ME类");  // dataSet.addValue(8000, "第三季度", "J2EE类");  // //添加第四季度数据  // dataSet.addValue(8000, "第四季度", "J2SE类");  // dataSet.addValue(2000, "第四季度", "J2ME类");  // dataSet.addValue(9000, "第四季度", "J2EE类");  // return dataSet;  // }  //  // //创建JFreeChart 对象  // public static JFreeChart createChart() {  // //通过ChartFactory创建JFreeChart  // JFreeChart chart = ChartFactory.createBarChart3D(  // "JAVA图书销量统计", //图表标题  // "JAVA图书", //横轴标题  // "销量（本）", //纵轴标题  // createDataSet(), //数据集合  // PlotOrientation.VERTICAL, //图表方向  // true, //是否显示图例标识  // false, //是否显示tooltips  // false); //是否支持超链接  // //背景图片  // Image image = null;  // //File file = new File("/image/test.jpg");  // try {  // //创建背景图片  // //image = ImageIO.read(file);  // image = ImageIO.read(CharUtil.class.getResource("/image/test.jpg"));  // } catch (IOException e) {  // e.printStackTrace();  // }  // //设置标题字体  // chart.getTitle().setFont(new Font("隶书",Font.BOLD,25));  // //设置图例类别字体  // chart.getLegend().setItemFont(new Font("宋体",Font.PLAIN,12));  // chart.setBorderVisible(true); //设置显示边框  // //实例化TextTitle对象  // TextTitle subTitle = new TextTitle("2009年Java类图书全国销量统计（J2SE、J2ME、J2EE）");  // //设置居中显示  // subTitle.setVerticalAlignment(VerticalAlignment.BOTTOM);  // chart.addSubtitle(subTitle); //添加子标题  // //获取绘图区对象  // CategoryPlot plot = chart.getCategoryPlot();  // plot.setForegroundAlpha(0.8F); //设置绘图区前景色透明度  // plot.setBackgroundAlpha(0.5F); //设置绘图区背景色透明度  // plot.setBackgroundImage(image); //设置绘图区背景图片  // //获取坐标轴对象  // CategoryAxis categoryAxis = plot.getDomainAxis();  // //设置坐标轴标题字体  // categoryAxis.setLabelFont(new Font("宋体",Font.PLAIN,12));  // //设置坐标轴标尺值字体  // categoryAxis.setTickLabelFont(new Font("宋体",Font.PLAIN,12));  // //设置坐标轴标题旋转角度  // categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.UP\_45);  // //获取数据轴对象  // ValueAxis valueAxis = plot.getRangeAxis();  // //设置数据轴字体  // valueAxis.setLabelFont(new Font("宋体",Font.PLAIN,12));  // //获取图片渲染对象  // BarRenderer3D renderer = new BarRenderer3D();  // renderer.setItemMargin(0.32); //设置柱子间的间距  // plot.setRenderer(renderer); //设置图片渲染对象  // return chart;  // }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*通过查询数据库并初始化数据集合\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //initPieData()方法用于创建数据集合对象，在此方法中，使用JDBCPieDataset类通过JDBC查询数据库获取数据集合对象。  //PiePlot类的setLabelGenerator()方法用于设置分类标签的格式，其参数为StandardPieSectionLabelGenerator对象，此对象的入口参数”{0}={2}”用于指定类别名称及所占有的百分比，{0}代表类别名称，{2 }代表百分比。  public static PieDataset initPieData() throws IOException {  Properties jdbcPros = PropertiesUtill.getpros("db.properties");  String driverName = jdbcPros.getProperty("jdbc.driverClass");  String url = jdbcPros.getProperty("jdbc.jdbcUrl");  String user = jdbcPros.getProperty("jdbc.user");  String password = jdbcPros.getProperty("jdbc.password");  JDBCPieDataset dataset = null; //JDBC数据集合  try {  //创建了JDBCPieDataset 对象后，可以通过executeQuery()方法查询数据库，其入口参数为String类型的SQL语句，执行此方法后，将返回拥有数据的数据集合对象。  dataset = new JDBCPieDataset(url, driverName, user, password); // 通过JDBC创建数据集合  String query = "select category,val from tb\_shop"; // SQL语句  dataset.executeQuery(query); // 查询并向数据集合中添加数据  dataset.close(); //关闭数据库连接  } catch (Exception e) {  e.printStackTrace();  }  return dataset;  }  //创建饼形图实例  public static JFreeChart createChart() throws IOException {  // 创建3D饼型图表  JFreeChart chart = ChartFactory.createPieChart3D(  "XX商城月销量统计", // 图表的标题  initPieData(), // 饼形图的数据集对象  true, // 是否显示图例  true, // 是否显示提示文本  false); // 是否生成超链接  chart.getTitle().setFont(new Font("隶书", Font.BOLD, 25)); //设置标题字体  chart.getLegend().setItemFont(new Font("宋体", Font.BOLD, 15)); //设置图例类别字体  PiePlot plot = (PiePlot) chart.getPlot(); // 获得绘图区对象  plot.setForegroundAlpha(0.5f); // 设置前景透明度  plot.setLabelFont(new Font("宋体", Font.PLAIN, 12)); // 设置分类标签的字体  plot.setCircular(true); // 设置饼形为正圆  // 设置分类标签的格式  plot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}={2}",  NumberFormat.getNumberInstance(),  NumberFormat.getPercentInstance()));  return chart;  }  } |

#### 2.1.2.8 解决中文乱码问题

1.由于JFreeChart组件的版本、操作平台、JDK的设置等因素，在使用JFreeChart组件时，可能出出现中文乱码的现象。遇到此问题时，可通过设置乱码文字的字体

进行解决，在此提供两种解决此问题的方法:

(1)设置主题样式

在制图前，创建主题样式并指定样式中的字体，通过ChartFactory的setChartTheme()方法设置主题样式，在指定制图样式后，ChartFactory对象创建的制

图对象将按此样式进行显示，图表中的文字将按指定的字体进行显示。

例如：

|  |
| --- |
| StandardChartTheme standardChartTheme = new StandardChartTheme("CN"); //创建主题样式  standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 20)); //设置标题字体  standardChartTheme.setRegularFont(new Font("宋体", Font.PLAIN, 15)); //设置图例的字体  standardChartTheme.setLargeFont(new Font("宋体", Font.PLAIN, 15)); //设置轴向的字体  ChartFactory.setChartTheme(standardChartTheme); //应用主题样式 |

通过上述代码设置主题样式后，再通过ChartFactory创建JFreeChart的对象，可以解决中文乱码问题。

(2)指定乱码文字的字体

此方法通过指定制图对象中的中文字体解决中文乱码问题，在图中任何用到中文的地方，都要对字体进行设置，此操作将涉及到JFreeChart对象、Plot对象、坐

标轴对象的属性设置。

例如：

|  |
| --- |
| JFreeChart chart = null;  …… // 省略部分代码  chart.getTitle().setFont(new Font("隶书",Font.BOLD,25)); // 设置标题字体  chart.getLegend().setItemFont(new Font("宋体",Font.BOLD,15)); // 设置图例类别字体  PiePlot plot = (PiePlot) chart.getPlot(); // 获得绘图区对象  plot.setLabelFont(new Font("宋体",Font.PLAIN,12)); // 设置分类标签的字体 |

不同的制图对象类型，其相关字体的设置可能存在差异。

# 9999下拉

Cloud boot 工具集不是框架