## 银行转账案例

### 案例需求

用户访问登录页面，在页面中输入用户名和密码点击登录，登录成功后

进入到主页中。如果登录失败，则在登录页面中提示用户用户名或密码

错误。在主页面中，用户在填写银行转账信息，包括:转账账户，转账账

户密码，转账金额，收款账号，收款人姓名。点击开始转账，完成转账。

但是有如下特殊需求:

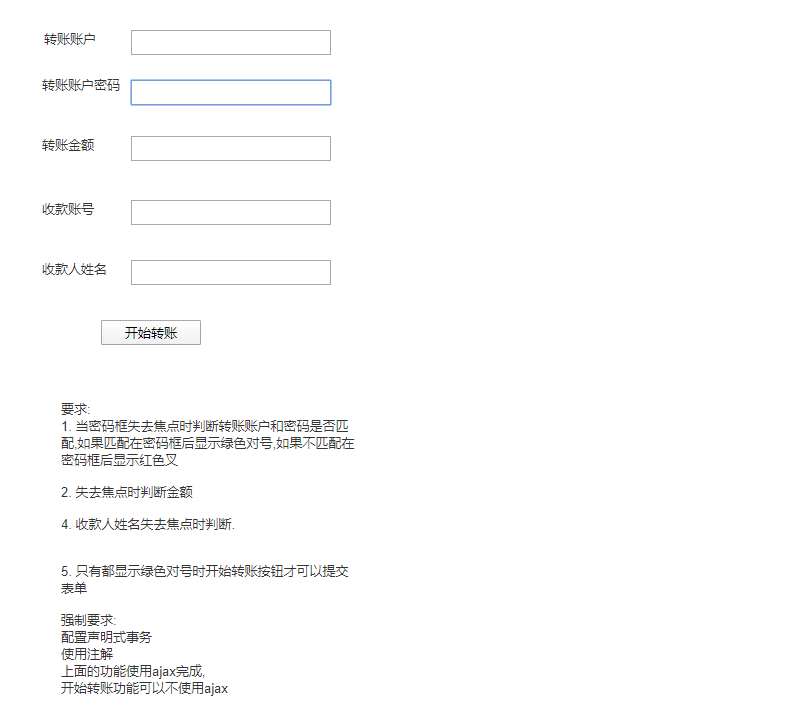
1. 需要用户的登录日志。
2. 当转账账户的密码失去焦点时，检验转账账户信息是否存在。
3. 在转账金额失去焦点时，校验金额是否正确。
4. 收款人姓名失去焦点时，校验收款人信息是否正确
5. 当所有的信息验证无误后，才能转账。

### 案例页面效果图

1. 登录页面效果图



1. 转账主页面效果图



### 功能分析

1. 创建登录页面
2. 用户登录以及登录失败提示语
3. 登录日志
4. 创建银行转账页面
5. 转账账户信息校验
6. 金额校验
7. 入账账户信息校验
8. 转账功能

### 数据库设计

1. 用户信息表: t\_user

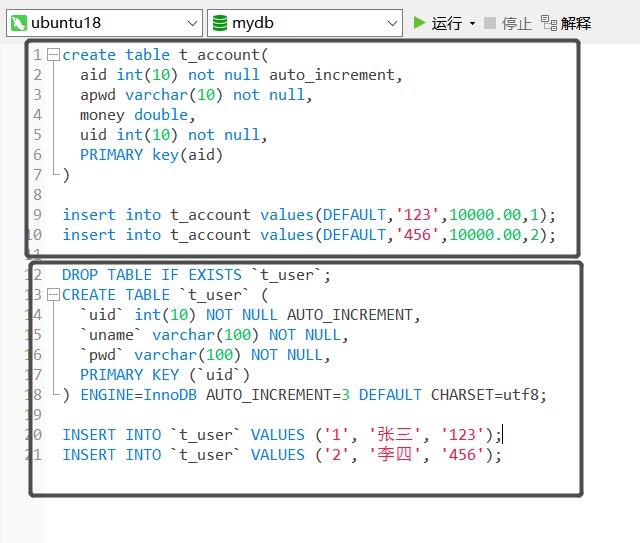
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 大小 | 约束 |
| uid | 用户编号 | int | 10 | 主键、非空 |
| uname | 用户名 | varchar | 100 | 非空 |
| pwd | 密码 | varchar | 100 | 非空 |

1. 银行账户信息表:t\_account

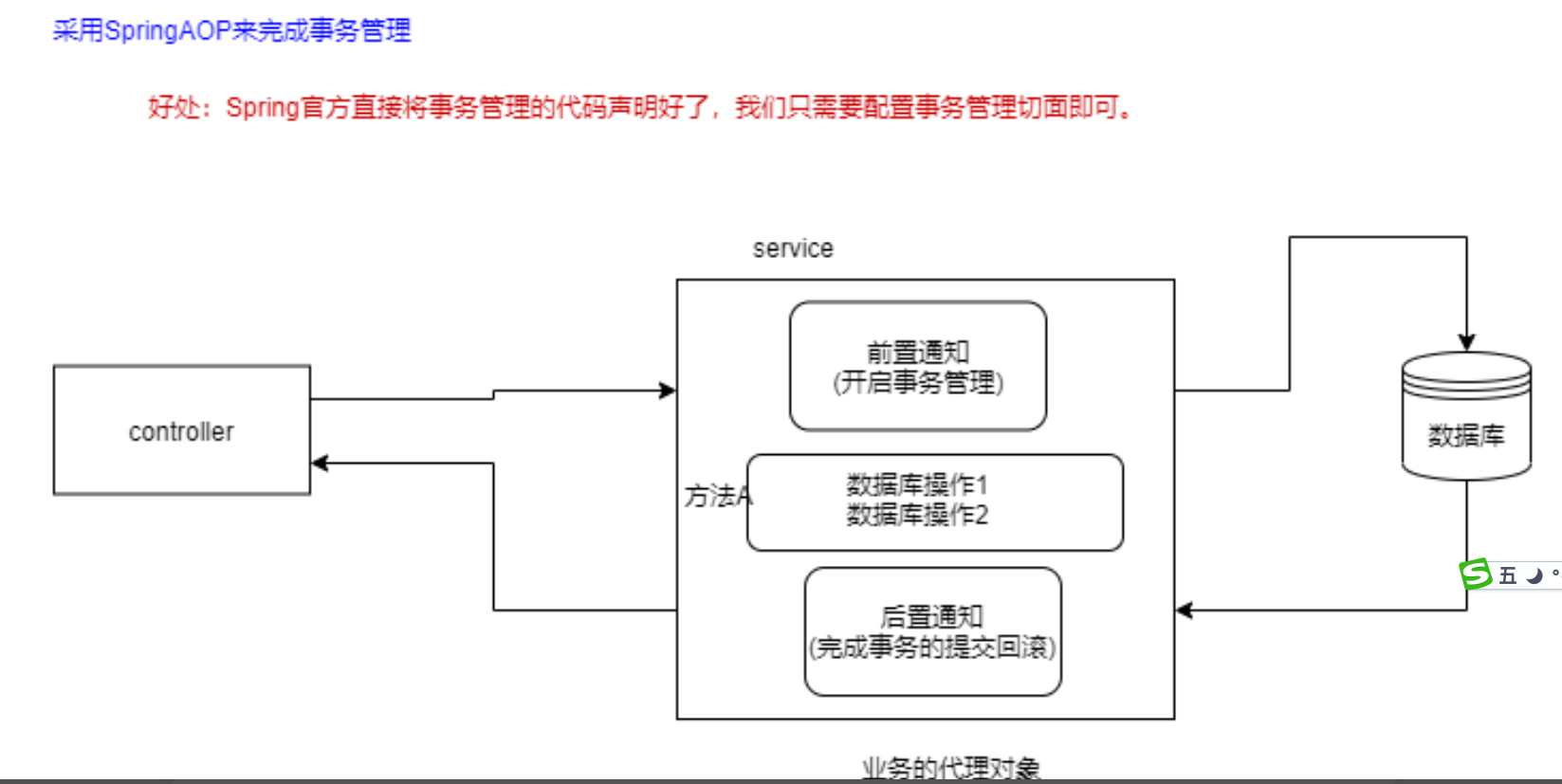
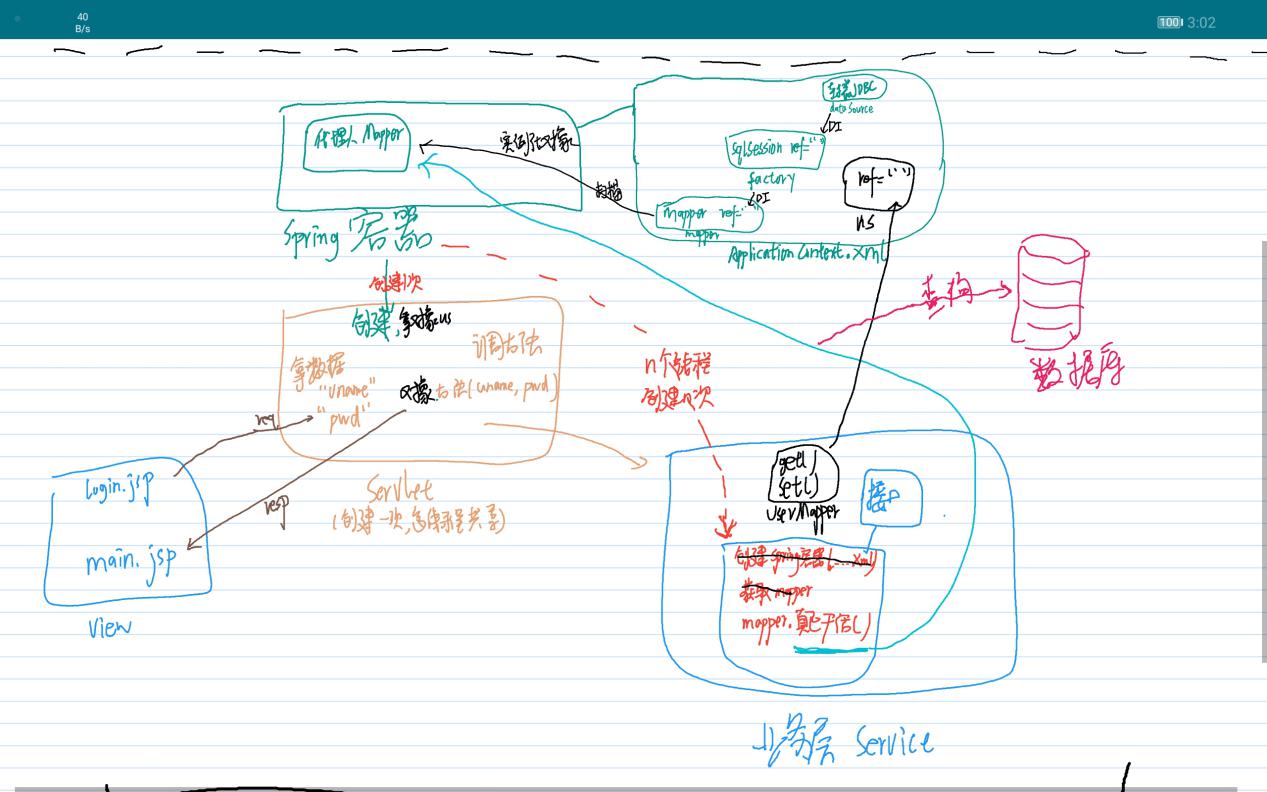
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 大小 | 约束 |
| aid | 账户编号 | int | 10 | 主键、非空 |
| apwd | 支付密码 | varchar | 100 | 非空 |
| money | 账户金额 | double |  |  |
| uid | 用户编号 | int | 10 | 非空 |

### 功能实现

#### 完成数据库的变现



#### 搭建Spring+Mybatis项目开发的环境



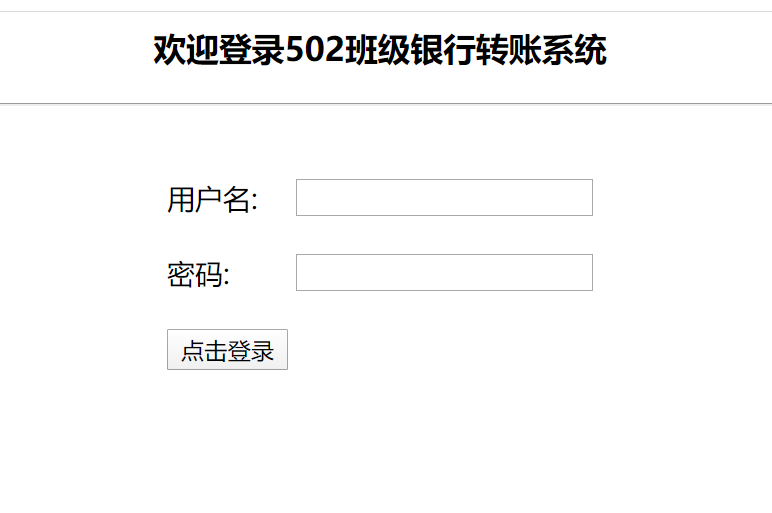
1. 创建web项目
2. 在web-inf下创建lib文件夹，并导入整合的jar包
3. 在src创建MVC的包结构

注意:将表的实体类文件创建

1. 在src下创建spring的配置文件
2. 在web.xml文件中配置Spring文件路径和监听器

#### 创建登录页面

效果图:

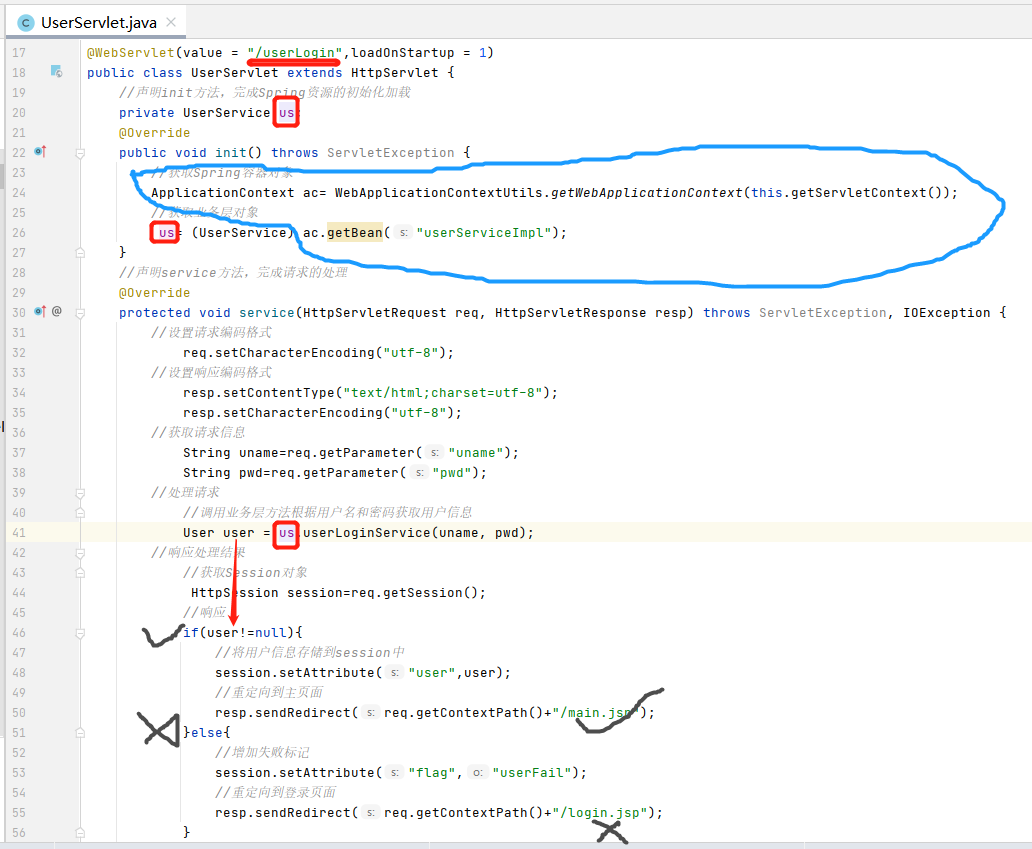


代码示例:

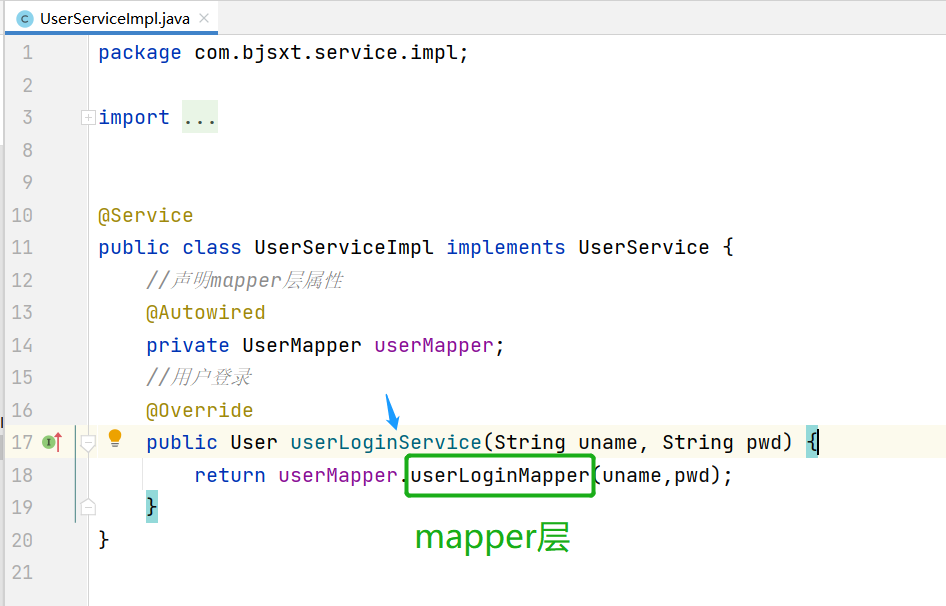


#### 完成登录功能

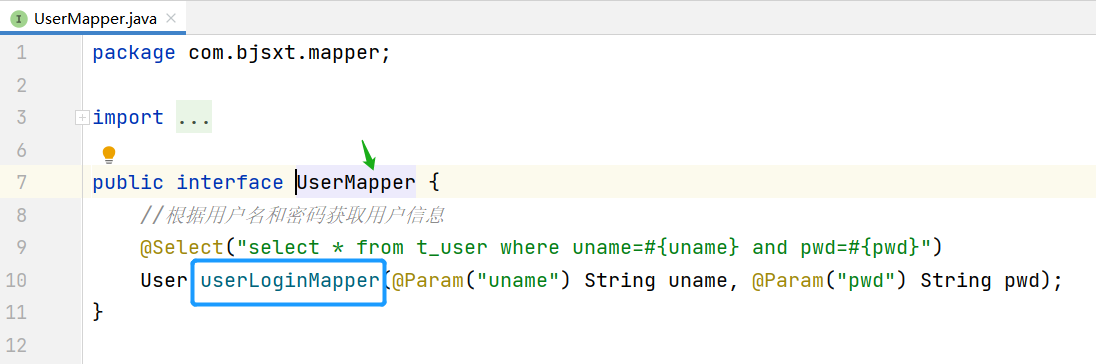
1. 创建UserServlet，处理登录请求



1. 创建UserService,处理登录请求



1. 创建UserMapper，根据用户名和密码获取用户信息



1. 在applicationcontext.xml文件中配置业务层bean

|  |
| --- |
| *<!--配置业务层bean-->* <**bean id="us" class="com.bjsxt.service.impl.UserServiceImpl"**>  <**property name="userMapper" ref="userMapper"**></**property**>  </**bean**> |

#### 创建main.jsp主页面

1. 页面示例图:



1. 代码示例:

|  |
| --- |
| *<%--  Created by IntelliJ IDEA.  User: zyp  Date: 2020/3/5  Time: 17:31  To change this template use File | Settings | File Templates. --%>* **<%** String path = request.getContextPath();  String basePath = request.getScheme() + **"://"** + request.getServerName() + **":"** + request.getServerPort() + path + **"/"**; **%>** <%@ **page contentType**="**text/html;charset=UTF-8**" **language**="**java**" %> <**html**> <**head**>  <**base href="<%=**basePath **%>"**/>  <**title**>Title</**title**>  *<%--引入jquery文件--%>* <**script type="text/javascript" src="js/j.js"**></**script**>  *<%--声明js代码域--%>* <**script type="text/javascript"**>  *//声明页面加载事件* $(**function** () {  *//创建颜色数组* **var** colors=[**"red"**,**"yellow"**,**"black"**,**"green"**,**"gray"**,**"blue"**,**"brown"**,**"darhorchid"**];  *//使用间隔执行* ***window***.setInterval(**function** () {  *//获取0-8的随机整数，含0不含8* **var** index=***Math***.floor(***Math***.random()\*colors.**length**);  *//设置标题颜色* $(**"#title"**).css(**"color"**,colors[index]);  },100)  })  </**script**> </**head**> <**body**>  *<%--网页的主题--%>* <**h3 align="center"**>  *<%--跑马灯效果--%>* <**marquee width=400 behavior=alternate direction=left align=middle**>  <**font color="r" id="title"**>欢迎**${**sessionScope.user.uname**}**登录502班级银行转账系统</**font**>  </**marquee**>  </**h3**>  *<%--水平线--%>* <**hr**>  *<%--银行转账表单--%>* <**div style="width**: 400**px**;**margin**: **auto**;**"**>  <**form action=""**>  <**table style="margin**: **auto**;**margin-top**: 30**px**;**" cellpadding="10px"**>  <**tr**>  <**td**>转账账户:</**td**>  <**td**>  <**input type="text" name="outId" value=""**>  </**td**>  </**tr**>  <**tr**>  <**td**>转账账户密码:</**td**>  <**td**>  <**input type="password" name="outPwd" value=""**>  </**td**>  </**tr**>  <**tr**>  <**td**>金额:</**td**>  <**td**>  <**input type="text" name="money" value=""**>  </**td**>  </**tr**>  <**tr**>  <**td**>收款账号:</**td**>  <**td**>  <**input type="text" name="inId" value=""**>  </**td**>  </**tr**>  <**tr**>  <**td**>收款人姓名:</**td**>  <**td**>  <**input type="text" name="inName" value=""**>  </**td**>  </**tr**>  <**tr**>  <**td colspan="2"**>  <**input type="submit" value="开始转账"**>  </**td**>  </**tr**>  </**table**>  </**form**>  </**div**> </**body**> </**html**> |

#### 完成转账账户信息的校验

1. 功能点需求:

用户在输入账户以及密码后，在密码失去焦点后，触发校验。如果正确则在密码框

后显示√，错误则显示X.

1. 功能点分析:

通过需求我们发现，校验的结果是在当前页面中继续显示的。相当于我们在当前

页面中显示新的响应结果 ，确定该功能使用ajax技术来实现。

1. 前台分析:
2. ajax请求的触发时机：在密码框失去焦点时触发
3. ajax请求的请求地址: Servlet的别名，checkAccount
4. ajax 请求的请求数据:账户和密码
5. ajax请求的响应数据:true或者false的字符串
6. 前台代码实现:

|  |
| --- |
| *//给密码框添加焦点事件，完成校验*  $(**function** () {  $(**"#outdPwd"**).blur(**function** () {  *//发起ajax请求* $.post(**"checkAccount"**,{**outId**:$(**"#outId"**).val(),**outPwd**:$(**"#outdPwd"**).val()，methodName=”checkOutInfo”},**function** (data) {  **if**(*eval*(data)){  $(**"#outSpan"**).**html**(**"√"**).css(**"color"**,**"green"**);  }**else**{  $(**"#outSpan"**).**html**(**"X"**).css(**"color"**,**"red"**);  }  })  }) }) |

1. 后台分析:
2. 创建Servlet，处理转账账户信息校验。checkAccount

注意:有，响应”true”,没有，响应”false”

1. 创建Service，处理账户信息校验。
2. 创建mapper，根据账户ID和密码获取账户信息。
3. 后台代码实现:

参照源码即可。

#### Servlet的优化

1. 问题:

目前我们每实现一个功能点，就需要在后台声明一个对应的Servlet进行

请求的处理。这样造成Servlet声明过多，代码繁琐。

1. 解决:

Servlet只声明一个，让所有的相关请求，都请求该Servlet。

在该Servlet中的service方法中，不做具体的请求处理。在

Service中，根据请求，调用对应的逻辑代码来处理请求即可。

类似：main方法的机制。

1. 实现:

在Servlet中将不同请求的处理逻辑代码封装成对应的逻辑功能方法

然后在service方法中，根据请求调用对应的逻辑方法处理请求。每次

请求都需要在请求数据中携带要请求的方法的方法名。后台service中

根据请求数据中的方法名调用对应的逻辑方法即可。

|  |
| --- |
| **package** com.bjsxt.controller; **import** com.bjsxt.pojo.Account; **import** com.bjsxt.service.CheckAccountService; **import** org.springframework.context.ApplicationContext; **import** org.springframework.context.support.ClassPathXmlApplicationContext; **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; @WebServlet(value = **"/checkAccount"**,loadOnStartup = 2) **public class** CheckAccountServlet **extends** HttpServlet {  *//声明业务层属性* **private** CheckAccountService **checkAccountService**;  *//重写init方法* @Override  **public void** init() **throws** ServletException {  *//获取Spring容器对象* ApplicationContext ac=**new** ClassPathXmlApplicationContext(**"applicationcontext.xml"**);  *//获取业务层对象* **checkAccountService**= (CheckAccountService) ac.getBean(**"checkAccountService"**);  }  *//重写service方法* @Override  **protected void** service(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {  *//设置请求编码格式* req.setCharacterEncoding(**"utf-8"**);  *//设置响应编码格式* resp.setCharacterEncoding(**"utf-8"**);  resp.setContentType(**"text/html;charset=utf-8"**);  *//获取请求数据中的方法名* String methodName=req.getParameter(**"methodName"**);  *//根据请求调用对应的逻辑代码* **if**(**"checkOutInfo"**.equals(methodName)){  checkOutInfo(req,resp);  }**else if**(**"checkMoneyInfo"**.equals(methodName)){  checkMoneyInfo(req,resp);  }**else if**(**"checInInfo"**.equals(methodName)){  checInInfo(req,resp);  }**else if**(**"transferInfo"**.equals(methodName)){  transferInfo(req,resp);  }**else**{  System.***out***.println(**"没有对应的逻辑方法:"**+methodName);  }  }  *//转账* **private void** transferInfo(HttpServletRequest req, HttpServletResponse resp) {  }  *//校验收款人信息* **private void** checInInfo(HttpServletRequest req, HttpServletResponse resp) {  }  *//校验金额* **private void** checkMoneyInfo(HttpServletRequest req, HttpServletResponse resp) {  }   *//声明方法：校验转账账户信息* **private void** checkOutInfo(HttpServletRequest req, HttpServletResponse resp) **throws** IOException {  *//获取请求信息* String outId=req.getParameter(**"outId"**);  String outPwd=req.getParameter(**"outPwd"**);  *//处理请求  //调用业务层方法* Account account = **checkAccountService**.checkOutAccountInfoService(outId, outPwd);  *//响应结果* resp.getWriter().write(account!=**null**?**"true"**:**"false"**);  }   } |

#### 校验金额

1. 功能点需求:

用户在金额框中输入要转账的金额信息后，失去焦点时，完成金额信息的校验。

余额足够则在金额框后显示√，否则显示X

1. 功能分析:

使用ajax技术。

1. 前台分析
2. 给金额框增加焦点事件
3. 发起ajax请求
4. 请求数据为金额，转账账户Id，后台逻辑方法名
5. 响应数据为true或者false的字符串

示例代码：

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*转账金额校验\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* $(**function** () {  $(**"#money"**).blur(**function** () {  *//发起ajax请求* $.post(**"checkAccount"**,{**outId**:$(**"#outId"**).val(),**money**:$(**"#money"**).val(),methodName:"checkMoneyInfo"},**function** (data) {  **if**(*eval*(data)){  $(**"#moneySpan"**).**html**(**"√"**).css(**"color"**,**"green"**);  }**else**{  $(**"#moneySpan"**).**html**(**"X"**).css(**"color"**,**"red"**);  }  })  })  }) |

1. 后台分析:功能参照源码。
2. 在CheckAccountServlet中声明金额校验逻辑方法，并在service方法

完成调用

1. 在CheckAccountService中新增金额校验的业务方法
2. 在CheckAccountMapper中新增金额校验的数据库操作

#### 收款人信息校验

1. 功能点需求:

用户在收款人姓名失去焦点时，校验收款人信息。存在，则在收款人姓名框后显示 √，不存在则显示X

1. 功能分析
2. 前台分析
3. 给收款人姓名框增加焦点事件
4. 发起ajax请求完成校验
5. 请求数据为收款人账户ID，收款人姓名，后台逻辑方法名
6. 响应数据 true或者false的字符串

代码示例:

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*收款人信息校验\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* $(**function** () {  $(**"#inName"**).blur(**function** () {  *//发起ajax请求* $.post(**"checkAccount"**,{**inId**:$(**"#inId"**).val(),**inName**:$(**"#inName"**).val(),**methodName**:**"checInInfo"**},**function** (data) {  **if**(*eval*(data)){  $(**"#inNameSpan"**).**html**(**"√"**).css(**"color"**,**"green"**);  }**else**{  $(**"#inNameSpan"**).**html**(**"X"**).css(**"color"**,**"red"**);  }  })  })  }) |

1. 后台分析
2. 在CheckAccountServlet中声明对应的逻辑方法完成校验
3. 在CheckAccountService中声明业务方法
4. 在CheckAccountMapper中声明数据库操作

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### 开始转账

1. 功能点需求:

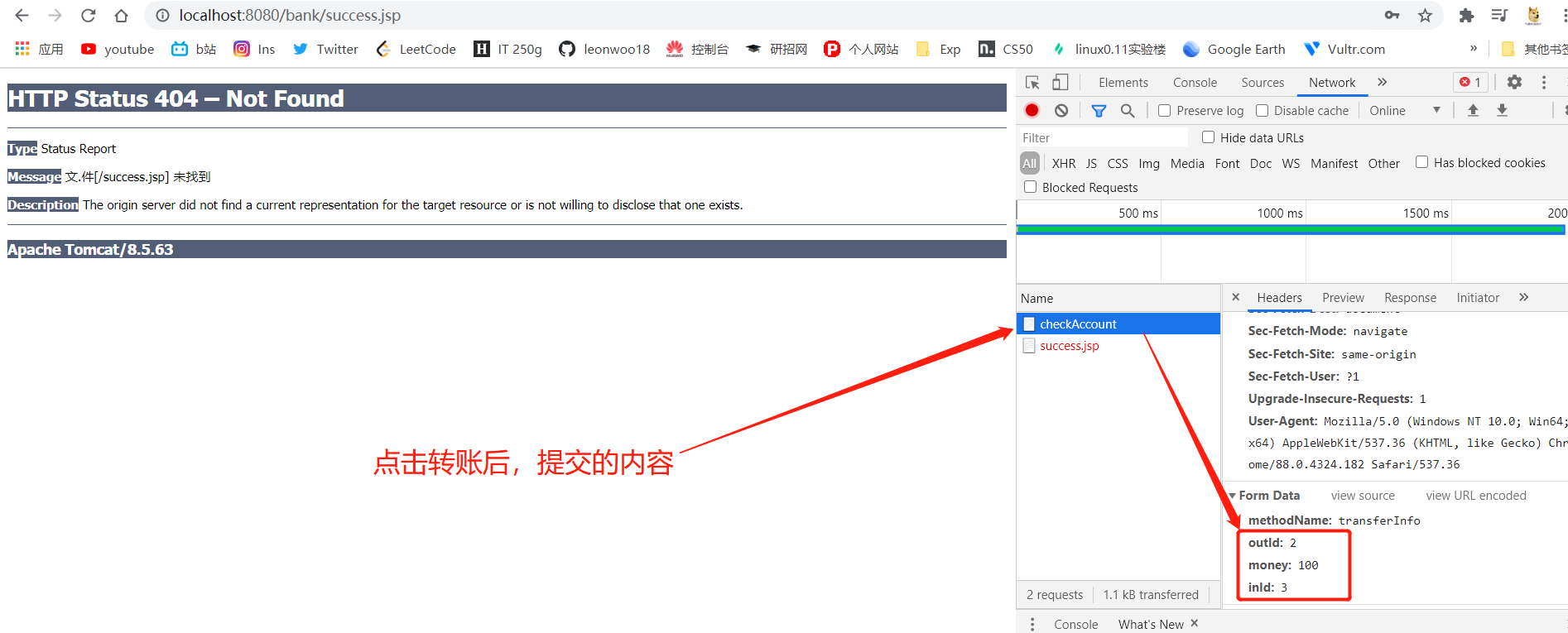
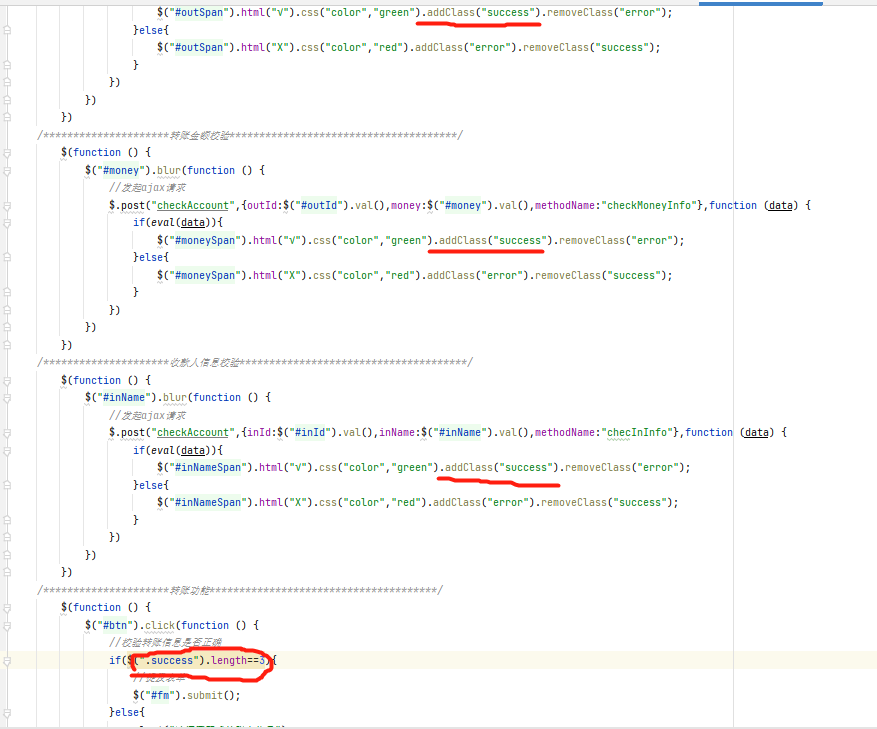
点击开始转账时，保证所有的转账信息是正确的，如果有错误的转账信息

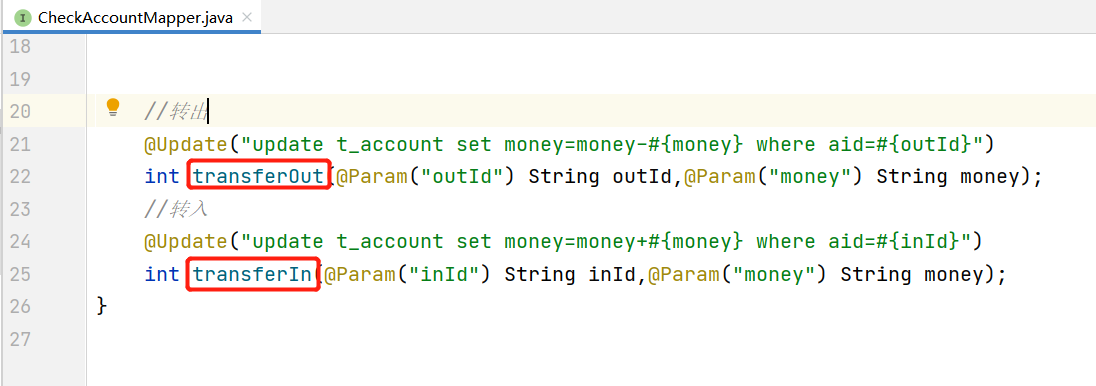
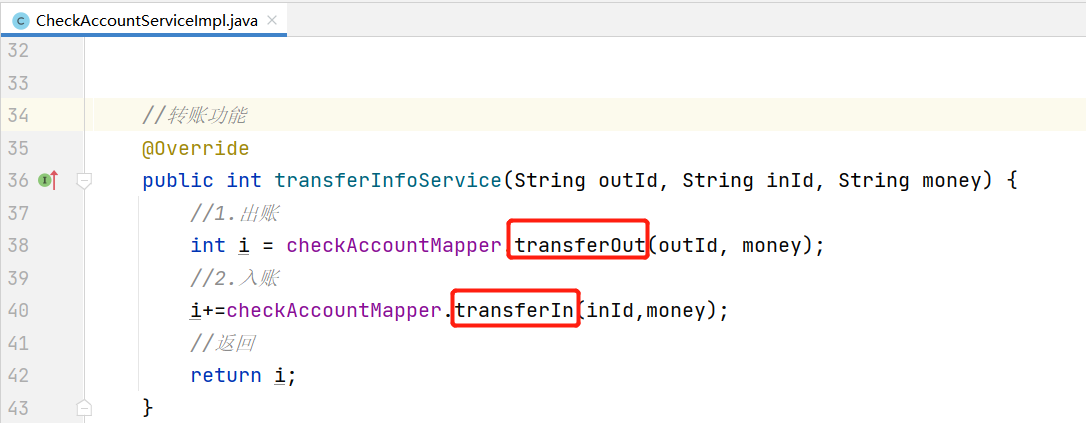
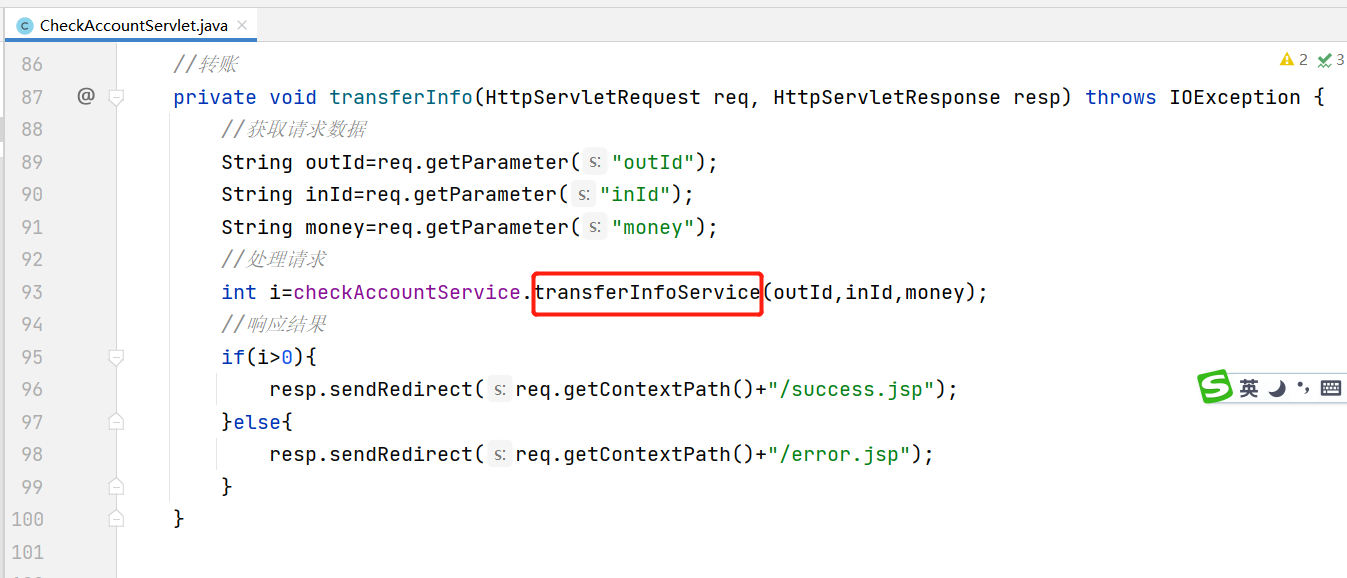
提示用户，请填写正确的信息。如果全部都是正确的，则直接发起转账请

求，完成转账。

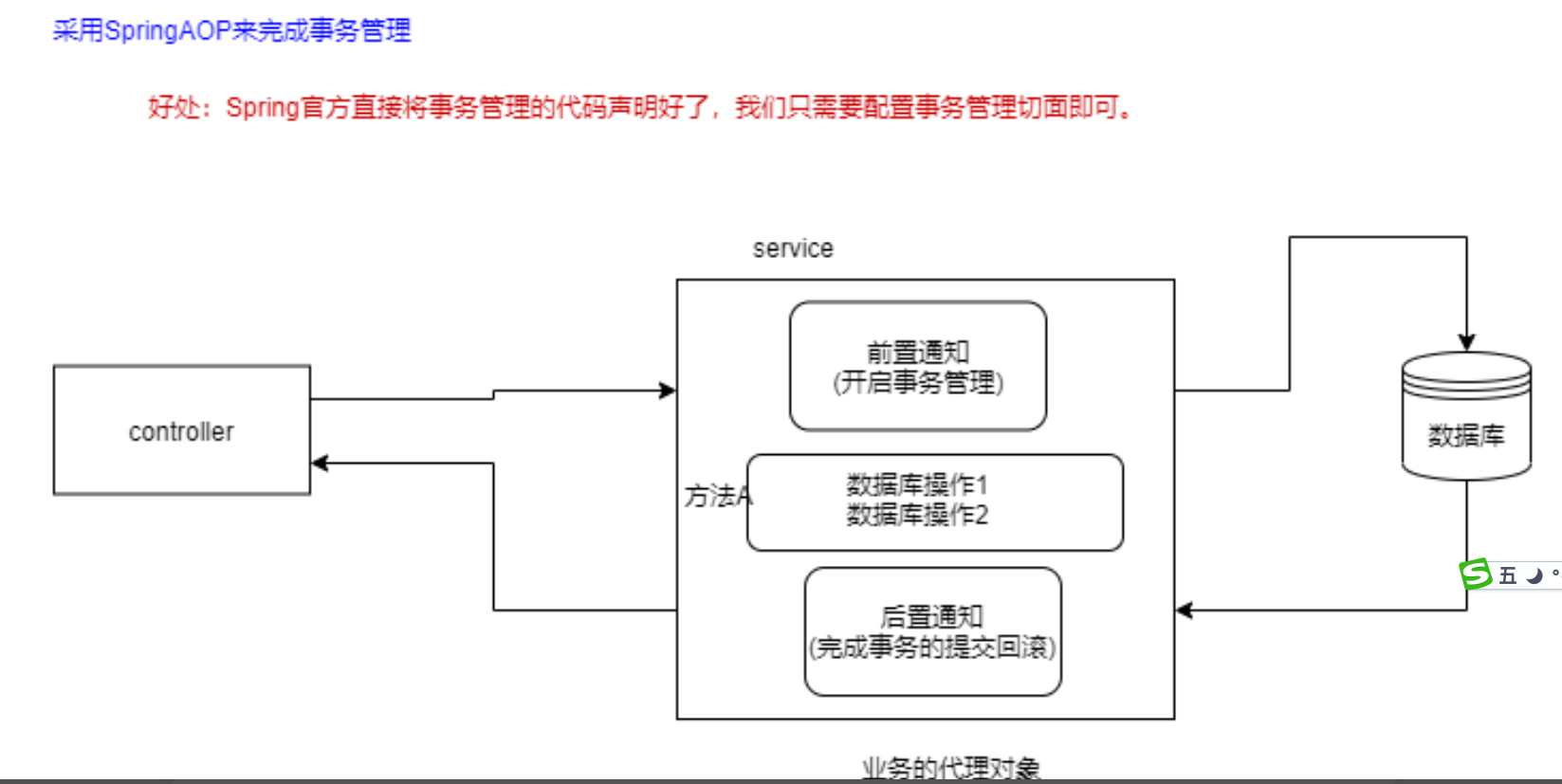
1. 功能分析:
2. 前台分析
3. 给转账按钮增加单击事件
4. 校验转账信息是否全部正确
5. 提交转账请求

代码示例:



1. 后台分析
2. 在CheckAccountServlet中声明转账的逻辑方法
3. 在CheckAccountService中声明转账的业务方法
4. 在CheckAccountMapper中完成转账的数据库操作

### 11.给转帐功能加上事务管理



### 在applicationcontext.xml文件中配置Spring的事务管理(重点掌握)



12.登录日志

1. 功能点需求:

我们的银行转账功能，在刚开始运营的时候，并没有用户登录的日志信息的记录。

后来，某天某个客户他的账号被登录了，但是给该客户又不知道是什么时间进行的

登录，造成调查有困扰，怎么办?在当前登录功能的基础上，增加登录日志，来记 录用户的登录信息。便于我们后期的排查。

1. 功能分析:

在保留当前登录功能的基础上，完成对登录功能的扩展。但是又不希望改变原有的

功能代码逻辑。考虑使用SpringAOP。

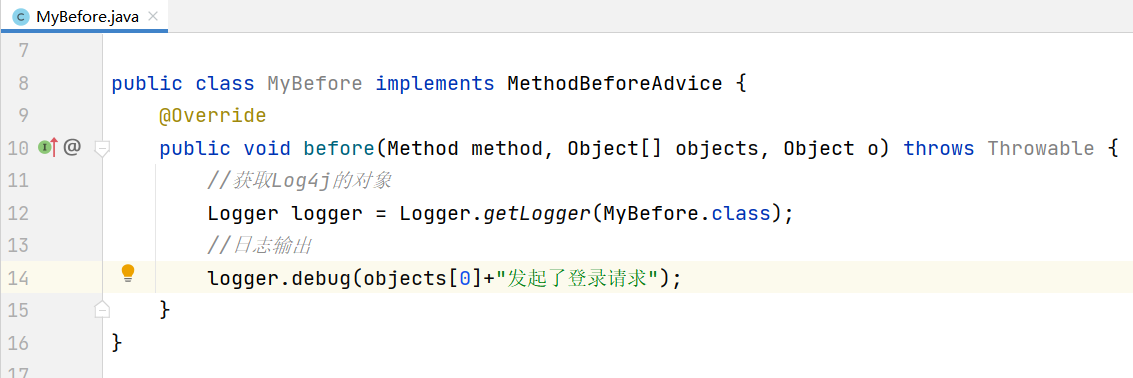
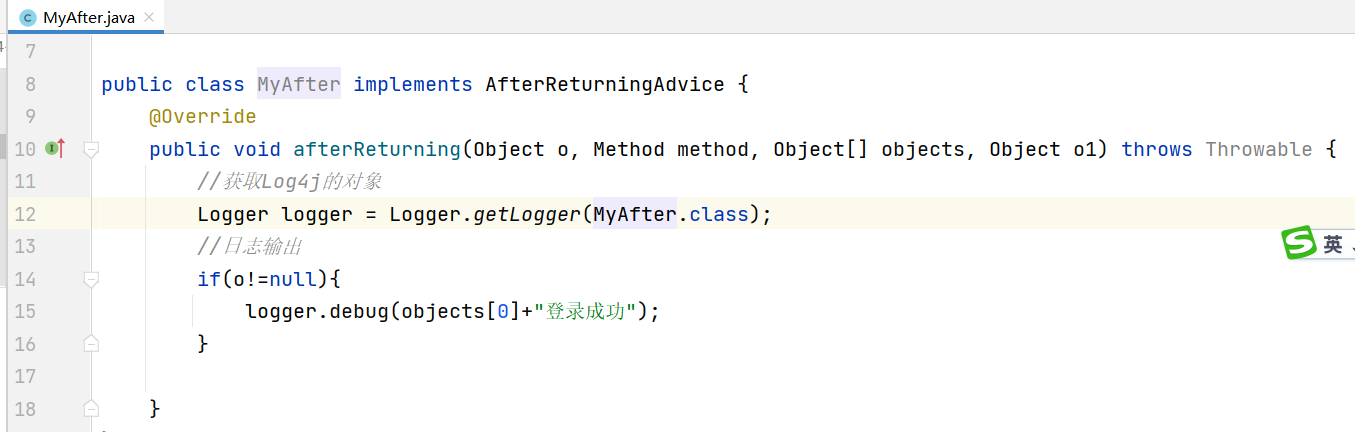
前置通知:在日志文件中输出XXX在XXX时间发起了登录请求

切点:登录的业务方法

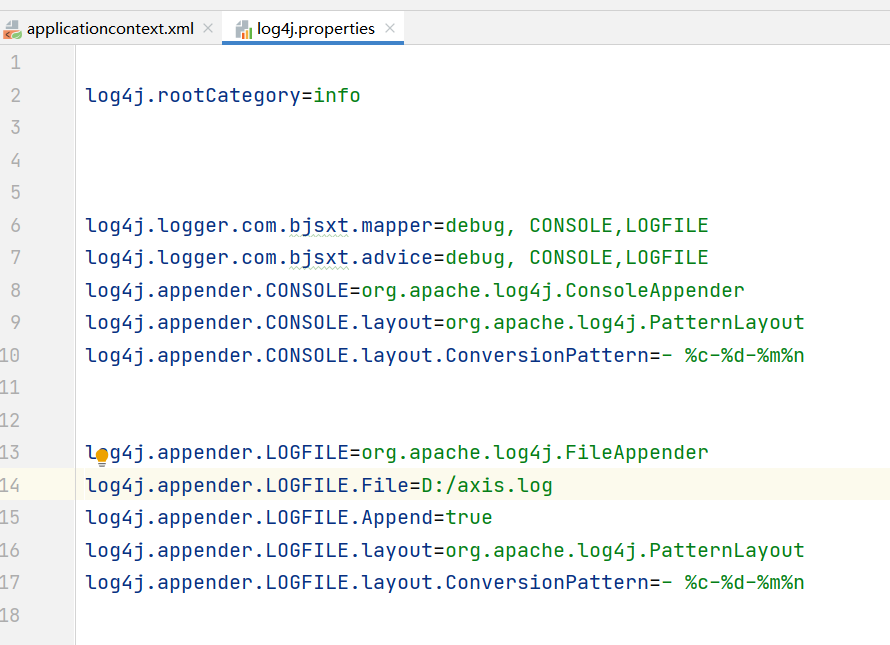
后置通知:在日志文件中输出XXX在XXX时间登录成功

建议使用SchemaBased方式实现。

1. 功能实现:
2. 在lib下导入AOP的jar包
3. 在src下创建com.bjsxt.advice的包
4. 在advice包中创建前置通知和后置通知



1. 在src下声明log4j.properties配置文件



1. 在applicationcontext.xml文件中配置通知bean以及组装规则