# 转账案例

【开发步骤】

1 转账的页面 收款人 付款人 金额 请求后台

2 servlet 接收参数 调用service（数据传递） 实现页面的控制

3 service 调用dao 事务应该在service

4 dao 接收参数 执行sql语句 操作数据库

【页面】

|  |
| --- |
| <body>  <h1>转账的页面</h1>  <form action=*"*${pageContext.request.contextPath }*/AccountServlet"* method=*"post"*>  <table border=*"1"* width=*"400"*>  <tr>  <td>付款人:</td>  <td><input type=*"text"* name=*"from"*/></td>  </tr>  <tr>  <td>收款人:</td>  <td><input type=*"text"* name=*"to"*/></td>  </tr>  <tr>  <td>转账金额:</td>  <td><input type=*"text"* name=*"money"*/></td>  </tr>  <tr>  <td colspan=*"2"*><input type=*"submit"* value=*"转账"*/></td>  </tr>  </table>  </form>  </body> |

【servlet】

|  |
| --- |
| //1.解析请求 获取页面请求数据  String from = request.getParameter("from");//付款人  String to = request.getParameter("to");//收款人  String money = request.getParameter("money");//金额  //2 调用service  try {  service.transfer(from, to, money);  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } |

【service】

|  |
| --- |
| public void transfer(String from ,String to ,String money) throws SQLException{  Connection conn = null ;  try{  //开启事务  //从当前线程中获取连接  conn = MyDbUtils2.*getConnection*();  conn.setAutoCommit(false);  //付款  dao.pay(from, money);  //int i = 1/0 ;  //收款  dao.inMoney(to, money);  conn.commit();  }catch(Exception e){  conn.rollback();  e.printStackTrace();  }  } |

【dao】

|  |
| --- |
| /\*\*  \* 付款  \* @throws SQLException  \*/  public void pay(String from , String money) throws SQLException{  //1 获取连接  Connection conn = MyDbUtils2.*getConnection*();  //获取操作sql语句的对象  String sql = "update user set money=money-? where name = ?" ;  PreparedStatement ps = conn.prepareStatement(sql);  //编译sql语句  ps.setString(1, money);  ps.setString(2, from);  //执行  ps.executeUpdate();  MyDbUtils.*release*(null, ps, null);  }  /\*\*  \* 收款  \* @throws SQLException  \*/  public void inMoney(String to , String money) throws SQLException{  //1 获取连接  Connection conn = MyDbUtils2.*getConnection*();  //获取操作sql语句的对象  String sql = "update user set money=money+? where name = ?" ;  PreparedStatement ps = conn.prepareStatement(sql);  //编译sql语句  ps.setString(1, money);  ps.setString(2, to);  //执行  ps.executeUpdate();  MyDbUtils.*release*(null, ps, null);  } |

【关于获取连接的两种方式】

1 原始的获取连接的方法

|  |
| --- |
| public static Connection getConnection() throws SQLException{  return DriverManager.*getConnection*(*url*, *username*, *password*);  } |

2 从当前线程中获取连接 （保证一个线程使用一个连接）

|  |
| --- |
| static ThreadLocal<Connection> *tl* = new ThreadLocal<>();  public static Connection getConnection() throws SQLException{  Connection conn = *tl*.get();  if(conn==null){  conn = DriverManager.*getConnection*(*url*, *username*, *password*);  //将连接放在线程里面  *tl*.set(conn);  }  return conn;  } |

【关于事务service层connection对象和dao的对象不一致问题】

1 在service层获取连接 向dao层传递

2 使用改造方法从线程中获取连接对象

# Ajax+json

【json数据格式简介】

|  |
| --- |
| //想描述user对象里面的属性 json 数据格式一 ： 简单的对象实体  var user\_1 = {"id":1,"name":"zhangsan","age":18} ;//代表一个对象 json  //alert(user.name);  //描述的是list 集合数据 数组  var list\_2 = [{"id":1,"name":"zhangsan","age":18},{"id":2,"name":"zhangsi","age":17}];  //alert(list\_2[1].name)  /\* for(i in list\_2){  alert(list\_2[i].name);  } \*/  /\* $.each(list\_2,function(i,n){  alert(n.name);  }); \*/  //3 map  var map = {"name":"lisi",  "age":19,  "user":{"id":1,"name":"zhangsan","age":18},  "list":[{"id":1,"name":"lisan","age":19},{"id":1,"name":"lier","age":21}]  }  //alert(map.list[1].name);  var str = '{"id":1,"name":"zhangsan","age":18}' ;  //将json格式的 字符串 转换成json对象 js里面的一个方法  var json = eval("("+str+")");//转换成json对象  //alert(json.name)  var str1 = '[{"id":1,"name":"zhangsan","age":18},{"id":2,"name":"zhangsi","age":17}]';  alert(str1[0].name); |

【ajax get+post】

|  |
| --- |
| //绑定点击事件  $("button").click(function(){  //触发ajax请求  /\* $.get("/WEB12/TestAjax",{"name":"吉吉"},function(data){  alert(data.name)  },"json"); \*/  //触发ajax请求  $.post("/WEB12/TestAjax",{"age":18},function(data){  alert(data.age)  },"json")  }) |

【get和post方式的参数】

|  |
| --- |
| 参数 1 url 请求的url  2 请求的数据 json格式  3 成功返回时的回调函数 一般携带响应的数据 data  4 接收的数据的数据格式 data json xml text |

【$.ajax】

|  |
| --- |
| $.ajax({  url:"/WEB12/TestAjax" ,//请求的url  data:{"name":"lisi"}, //携带的数据  type:"post",//请求的方式  success:function(data){//成功时的回调函数 string json jsp片段  alert(data.name);  },  //ajax不支持畸形的json格式的数据{‘name’:’lisi’}这种就是畸形  error:function(data){//失败时的回调函数  alert("服务器异常");  },  dataType:"json"//接收数据的处理格式  }); |

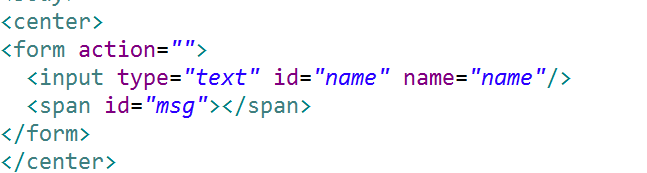
## 用户名是否存在校验案例

【需求分析】

|  |
| --- |
| 1 写一个页面 form <input> <span>  2 给文本框绑定一个时区焦点的事件 .blur()  1) 获取当前的文本框的值  2）发送ajax请求到服务器 携带 文本框的值 {name:$(“#name”).val()}  3)成功接收 “此用户名已经被注册”  4）获取span 将后台返回的数据放span html();  3 servlet 接收参数 name 调用service 判断返回值 有值响应数据“此用户名已经被注册”  4 service 调用dao  5 根据name查找数据库 返回值（User） |

## **【代码实现】**

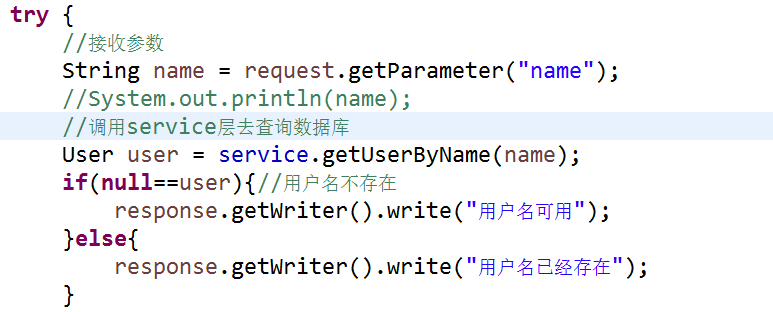
**1 页面**



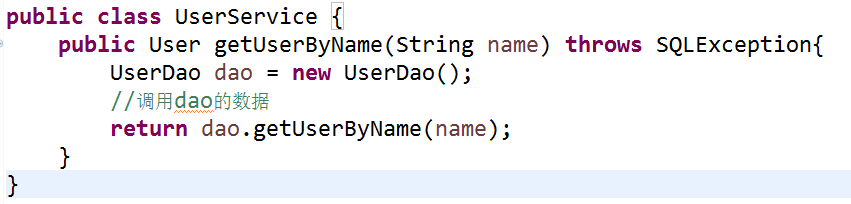
2 js代码



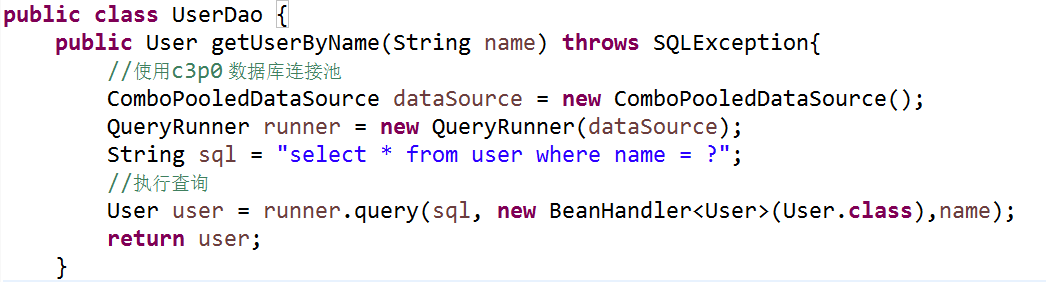
【web层】



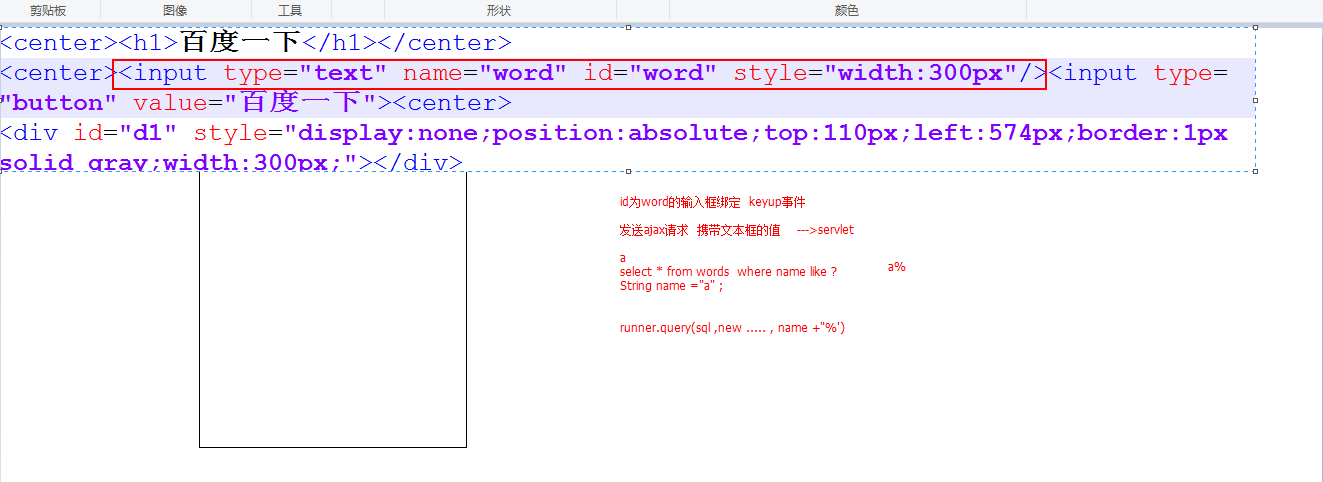
【service层】



## **【dao层】**



## 输入框提示案例



# 商品的前台页面展示

# 商品的详情页面完成添加购物车功能

# 商品的后台管理

# 支持商品的查询 ，删除 ，添加，修改 ，状态