# JDBC操作

**public** **class** JdbcTest **extends** HttpServlet{

**private** String driver = "com.mysql.jdbc.Driver";

**private** String url = "jdbc:mysql://127.0.0.1:3306/bookdb?useUnicode=true&characterEncoding=UTF-8";

**private** String sql = "select \* from user";

@Override

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

Connection con = **null**;

ResultSet result = **null**;

resp.setContentType("text/html;charset=UTF-8");

PrintWriter out = resp.getWriter();

**try** {

//1、注册

Class.*forName*(driver);

//2、获取连接

con = DriverManager.*getConnection*(url, "root", "123456");

//3、获取Statement

Statement state = con.createStatement();

//4、执行操作

result = state.executeQuery(sql);

**while**(result.next()) //显示结果集的内容

{

String name = result.getString("name");

System.***out***.println(name);

out.println(name);

}

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

@Override

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

doGet(req, resp);

}

}

配置web.xml

# 2.发送数据：

//JSON数据格式

//JSON里面对象【键值对】{}、数组【单个的数据】[]

String json = "";

**while**(result.next()) //显示结果集的内容

{

User user = **new** User();

user.setId(result.getInt("id"));

user.setName(result.getString("name"));

user.setNickname(result.getString("nickname"));

user.setPassword(result.getString("password"));

user.setSex(result.getInt("sex"));

user.setAge(result.getInt("age"));

user.setUsertype(result.getInt("usertype"));

Gson gson = **new** Gson();

json = gson.toJson(user);

out.println(json);

}

# 3.JS脚本：

<script type=*"text/javascript"*>

**function** getData() {

//Jquery:这里只使用JQuery的标志和ajax方法

//document.getElementById('id')==>$('id')

//{}:对象：键值对

$.ajax({

type:"post",

url:"http://localhost:8080/First/jdbc",

async:**true**,//是否异步请求

dataType:"json",//返回的数据类型

success:**function**(data){//当请求成功的时候调用的方法

$.each(data,**function**(n,value){//value:是每次的循环的值

$("tbody").append('<tr><td>'+value.id+'</td><td>'+value.name+'</td><td>'+value.nickname+'</td><td>'+value.password+'</td><td>'+value.age+'</td><td>'+value.sex+'</td><td>'+value.usertype+'</td></tr>');//追加内容

})

},

error:**function**(){

alert("请求失败");

}

});

}

getData();

</script>

## 修改的地方：

//5、发送到客户端

//JSON数据格式

//JSON里面对象【键值对】{}、数组【单个的数据】[]

String json = "";

List<User> list = **new** ArrayList<User>();

Gson gson = **new** Gson();

**while**(result.next()) //显示结果集的内容

{

User user = **new** User();

user.setId(result.getInt("id"));

user.setName(result.getString("name"));

user.setNickname(result.getString("nickname"));

user.setPassword(result.getString("password"));

user.setSex(result.getInt("sex"));

user.setAge(result.getInt("age"));

user.setUsertype(result.getInt("usertype"));

list.add(user);

}

json = gson.toJson(list);

out.println(json);

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

} **catch** (SQLException e) {

e.printStackTrace();

}

<div class=*"container"*>

<table class=*"table table-bordered"* id=*"table"*>

<thead>

<tr>

<th>ID</th>

<th>名字</th>

<th>昵称</th>

<th>密码</th>

<th>性别</th>

<th>年龄</th>

<th>用户类型</th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

# 4.JavaBean

* **JSP页面中与JavaBean有关的标记有3个：**
  + **<jsp:useBean>**
  + **<jsp:setProperty>**
  + **<jsp:getProperty>**
* **<jsp:useBean>声明一个具有一定生存范围及一个唯一id的JavaBean的实例，JSP页面通过id来识别JavaBean，并可通过id.method类似的语句来操作JavaBean。**
  + **例如，下面的标记在应用程序作用域中，声明了类型 Student、id为s1的 bean：**
    - ***<jsp:useBean id="s1" class=" Student " scope="application"/>***
  + **<jsp:getProperty>用来返回一个已被创建的bean组件的属性值。用法如下：**
    - ***<jsp:getProperty name="beanId" property="propertyName" />***
  + **<jsp:setProperty>用来设定一个已被创建的bean组件的属性值，用法如下：**

***<jsp:setProperty name="beanId" property="propertyName" value="propertyValue"/>***

* **除了利用<jsp:setProperty>标记和直接调用JavaBean对象的方法来设置JavaBean属性外，还可以利用request对象中的参数来设置JavaBean属性。**
* **语法如下，**
  + <JSP:setProperty name="beanId" property="propertyName" param="paramName"/>
  + 注意，如果参数值为空值，那么对应的属性值不会被设定。另外，不能在同一个“<JSP:setProperty>”中同时使用param和value参数 。

# 5.改造表格：使用bootstrap table

修改表格如下：

<div class="container">

<table class="table table-bordered" id="table" data-toggle="table">

<caption>明星参数</caption>

</table>

</div>

在表格前面同级增加如下：

<div class="container">

<div id="toolbar" class="btn-group">

<button id="btn\_add" type="button" class="btn btn-default">

<span class="glyphicon glyphicon-plus" aria-hidden="true"></span>新增

</button>

<button id="btn\_delete" type="button" class="btn btn-default">

<span class="glyphicon glyphicon-minus" aria-hidden="true"></span>删除

</button>

<button id="btn\_save" type="button" class="btn btn-default">

<span class="glyphicon glyphicon-save" aria-hidden="true"></span>保存

</button>

</div>

</div>

主页面使用：

<script type="text/javascript">

var oTable = new TableInit('http://localhost:8080/m/');

oTable.Init();

</script>

# 5.编写book sevlet

配置web.xml:

在web.xml中增加下面配置

<servlet>

<servlet-name>book</servlet-name>

<servlet-class>com.jdbc.JdbcBook</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>book</servlet-name>

<url-pattern>/book</url-pattern>

</servlet-mapping>

运行：



# 6.改造

修改JdbcBook servlet：

Map map = **new** HashMap<>();

map.put("total", "200");

map.put("rows", list);

json = gson.toJson(map);

out.println(json);

修改index.html:

改地址：

var oTable = new TableInit('http://localhost:8080/First/');

$('#table').bootstrapTable({

url: t\_url+'booksPage.do', //请求后台的URL（\*）

修改表格样式：

*.table* {

table-layout: *fixed*;

}

*.table* **td***:hover* {

overflow: *visible*;

white-space: *normal*;

}

*.table* **td** {

word-wrap: *break-word*;

overflow: *hidden*;

text-overflow: *ellipsis*;

white-space: *nowrap*;

}