哈尔滨工业大学计算机科学与技术学院

2017 年秋季学期《软件工程》

Lab 2：MVC 编程与云平台部署

姓名	学号	联系方式
马玉坤	1150310618	mayukuner@126.com/18845895386
实验结果在公共云平台的访问地址（URL）		http://library.qwertier.cn/library

目 录

1. 实验要求.....1
2. 开发环境配置.....1
2.1. 在 Eclipse 中配置 Struts2.....1
2.2. 在 Eclipse 中配置 MySQL.....1
2.3. 在 Eclipse 中配置 Tomcat.....2
3. 图书 SaaS 设计.....4
3.1. Web.xml.....4
3.2. Struts.xml.....4
3.3. Action 类.....6
3.4. 前端 Web 页面.....6
3.5. 各 Action/前端页面之间的调用和消息传递关系.....7
4. 图书 SaaS 核心代码.....8
4.1. 按作者查询.....8
4.2. 展示图书详细信息.....9
4.3. 删除图书.....13
4.4. 新增图书/作者(可选).....14
4.5. 更新图书信息(可选).....16
4.6. 数据库连接与访问.....19
5. 图书 SaaS 的云平台部署.....19
5.1. 所选定的公共云平台.....19
5.2. 部署步骤.....20
5.3. 外在访问结果.....20
6. 小结.....22

1. 实验要求

1.1. 掌握 MVC 架构下的 SaaS 开发的基本流程和技术

a) 开发环境：JAVA+Struts 2+Eclipse+Tomcat+MySQL+SAE(或其他云平台)

b) 在 Eclipse 中配置 Struts 开发环境，开发一个小型 SaaS ，在 Web 页面中对数据库中的数据进行 CRUD 操作；

c) 在 Struts 基础上，可以补充使用某些前端框架(如 Bootstrap, jQuery, node.js 等)或数据库访问框架(如 Hibernate 等)。

1.2. 使用 PaaS 云平台（如 SAE）或者 IaaS 云平台（如阿里云），部署在自选的公共云计算平台上并对外发布，模拟用户访问。

2. 开发环境配置

2.1. 在 Eclipse 中配置 Struts2

由于本人使用的是 Maven 管理依赖，因此只需要在项目的 pom.xml 文件中加入 struts 的依赖项，然后使用 maven install 即可。

```
<dependency>
    <groupId>org.apache.struts</groupId>
    <artifactId>struts2-core</artifactId>
    <version>2.5.13</version>
</dependency>
```

2.2. 在 Eclipse 中配置 MySQL

由于本人使用的是 Maven 管理依赖，因此只需要在项目的 pom.xml 文件中加入 mysql-connector-java 的依赖项，然后使用 maven install 即可。

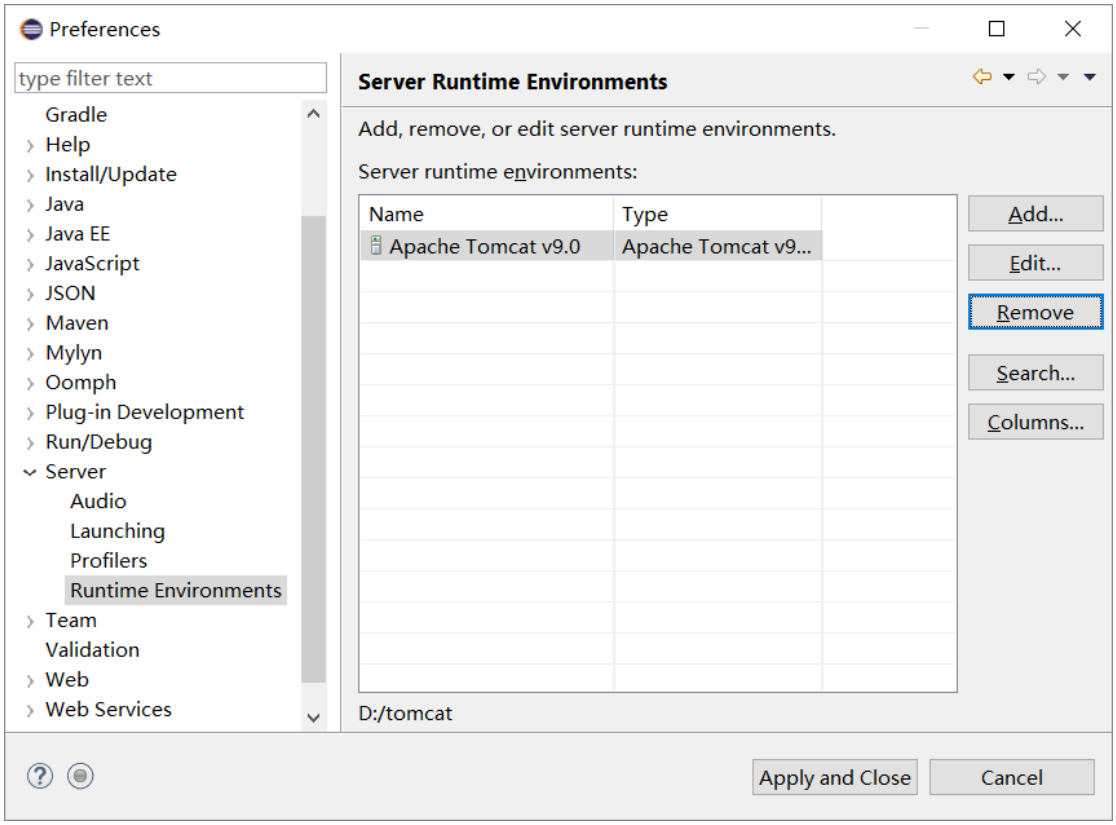
```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.44</version>
</dependency>
```

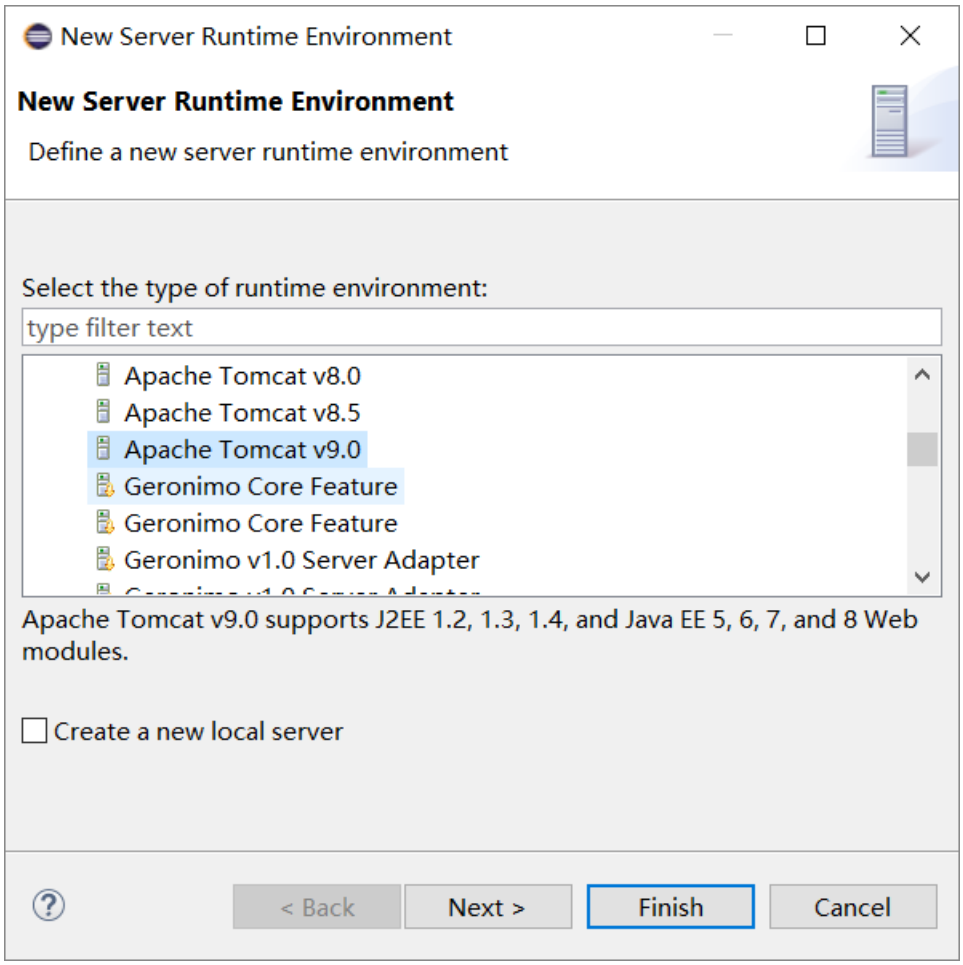
此外， 在配置 MySQL 时，需要注意编码问题。为了方便，我将所有用到的编码全部设置成了 UTF8 编码。通过修改 mysql 根目录下的 my.ini。添加下列代码：

```
[mysql]
default-character-set=utf8
[mysqld]
character-set-server=utf8
```

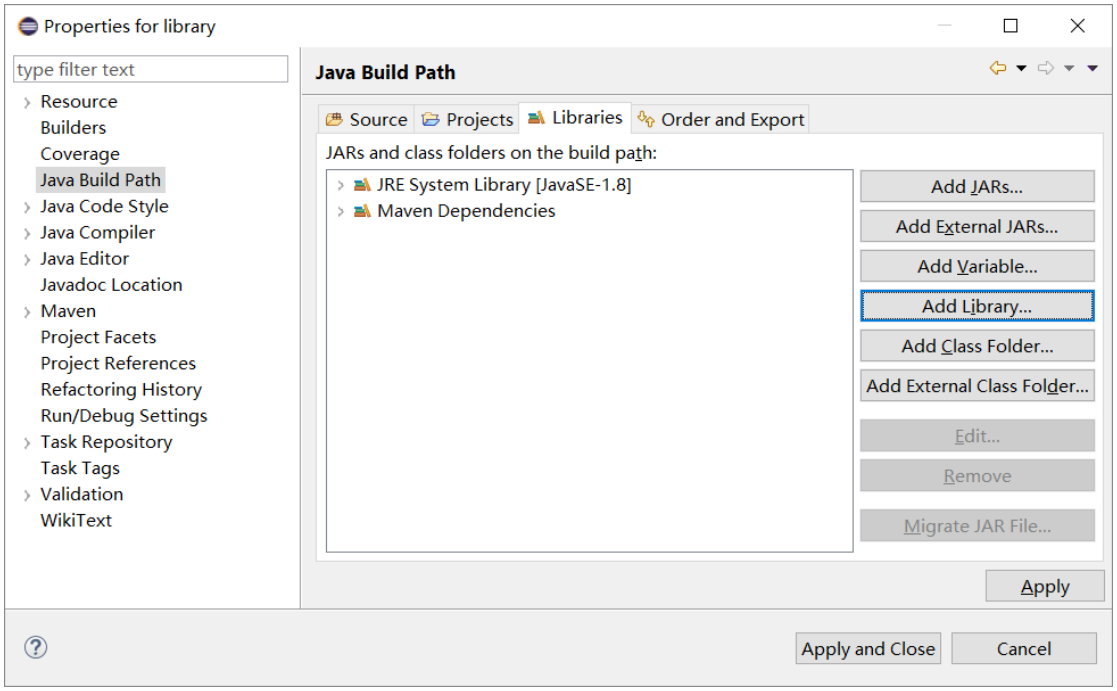
2.3. 在 Eclipse 中配置 Tomcat

首先打开 Window-Preferences，然后找到 Server-Runtime Environments 选项卡，点击 Add，添加 Tomcat 9.0 环境。





之后，需要在项目属性窗口中，找到 Java Build Path，把 Apache Tomcat 的运行环境加入项目的 Build Path 中。



点击 library，然后点击”Add Library”，添加”Server Runtime”-“Apache Tomcat v9.0”，即可。

3. 图书 SaaS 设计

3.1. Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  id="WebApp_ID" version="3.1">
  <display-name>library</display-name>
  <welcome-file-list>
      <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <filter>
      <filter-name>struts2</filter-name>

  <filter-class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExec
uteFilter</filter-class>
  </filter>
  <filter-mapping>
      <filter-name>struts2</filter-name>
      <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

welcome-file: 默认的首页文件，在本项目中为 index.jsp。
filter-class: 使用的用来处理网页请求的 java 类名，此处为 org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter，即 struts 框架中来处理服务器请求的类。

3.2. Struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 2.3//EN" "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
    <constant name="struts.devMode" value="true" />
    <constant name="struts.action.excludePattern" value="/static/.*?" />
    <package name="library" extends="struts-default, json-default">
        <default-action-ref name="index" />
```

```
        <action name="index">
            <result>/index.jsp</result>
        </action>
        <action name="logout">
            <result>/index.jsp</result>
        </action>
        <action name="login">
            <result>/queryBook.jsp</result>
        </action>
        <action name="queryBook">
            <result>/queryBook.jsp</result>
        </action>
        <action name="tableBook"
class="cn.qwertier.library.TableBookAction">
            <result>/tableBook.jsp</result>
        </action>
        <action name="dataTableBook"
class="cn.qwertier.library.DataTableBookAction">
            <result type="json" />
        </action>
        <action name="addBook">
            <result>/addBook.jsp</result>
        </action>
        <action name="getAuthorList"
class="cn.qwertier.library.GetAuthorListAction">
            <result type="json" />
        </action>
        <action name="getBook" class="cn.qwertier.library.GetBookAction">
            <result type="json" />
        </action>
        <action name="modifyBook"
class="cn.qwertier.library.ModifyBookAction">
            <result type="json" />
        </action>
        <action name="addAuthor"
class="cn.qwertier.library.AddAuthorAction">
            <result type="json" />
        </action>
    </package>
</struts>
```

`<constant name="struts.action.excludePattern" value="/static/.*?" />`用来处理静态文件，过滤/static/开头的请求。

`<package name="library" extends="struts-default, json-default">`需要注意的是

json-default，引入 json-default 是为了使用 struts-json 插件更方便地处理返回值为 json 数据的请求，struts-json 在 pom.xml 中已引入。

3.3. Action 类

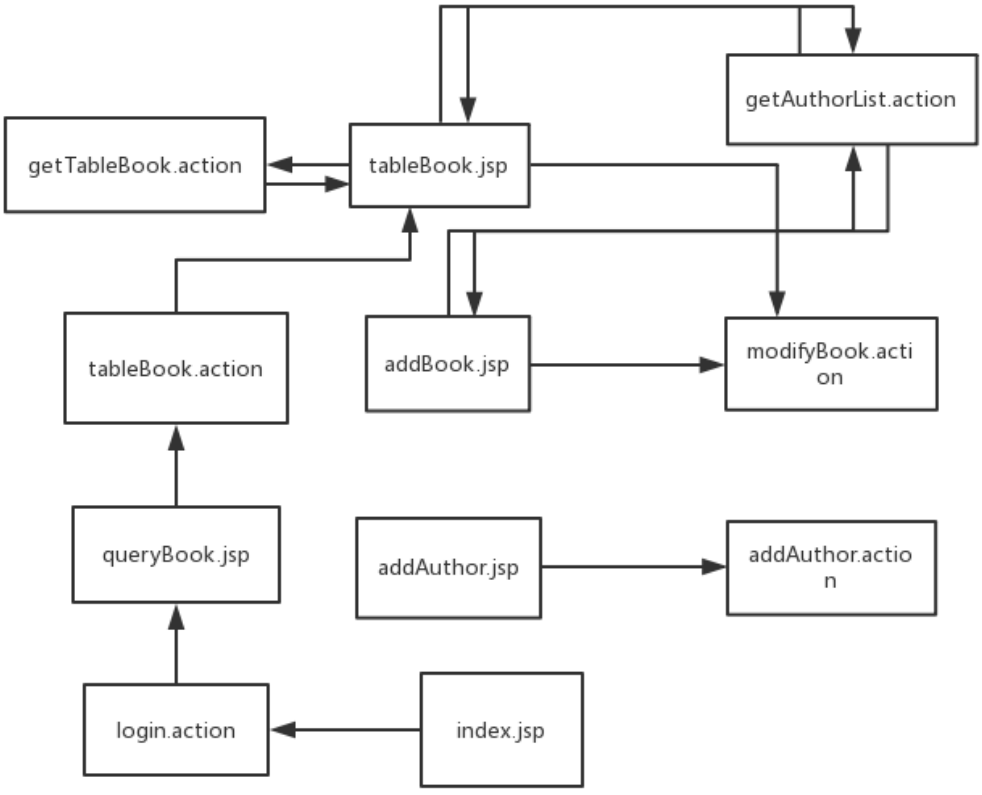
序号	Action 类名	作用	操作列表	操作说明
1	*TableBookAction*	获得用户提交的用来搜索图书信息的过滤数据，并提交给网页	*execute()*	将用户用来搜索图书信息的“作者名”、“isbn”等信息传给前端 jsp
2	*DataTableBookAction*	获得前端书籍展示表格发送的查询请求和查询关键词（如 ISBN、作者信息），处理后返回前端页面。	*execute()*	获得查询参数，返回查询的图书列表（json 格式）。
3	*GetAuthorListAction*	将服务器中的作者列表发送给前端	*execute()*	发送作者列表(json 格式)。
4	*GetBookAction*	将单本图书信息发送给前端	*execute()*	获取 isbn，并将对应书籍的图书信息发送给前端(json 格式)。
5	*ModifyBookAction*	处理修改、新建和删除图书信息的请求	*execute()*	获取图书的信息，并按照 isbn 号插入新书、修改图书信息，或者删除图书信息，使用 json 返回操作结果
6	*AddAuthorAction*	处理向数据库中添加作者的请求	*execute()*	获取作者的详细信息，并在数据库中添加对应的作者信息

3.4. 前端 Web 页面

序号	页面名	作用	页面核心元素 (form)	Form 对应的 action name	Form 中提交的数据项
1	index.jsp	登录入口，网站首页	login-form	login	用户名、密码
2	queryBook.jsp	查询数据信息的入口	query-book-form	tableBook	作者姓名、书名、出版社、出版日期（范

					围）
3	tableBook.jsp	使用表格显示查询的图书信息	show-author-form	无	无
			modify-book-form	modifyBook	ISBN（用户无法修改）、作者、出版社、出版日期、价格
4	addBook.jsp	添加图书信息	add-book-form	modifyBook	ISBN、作者、出版社、出版日期、价格
5	addAuthor.jsp	添加作者信息	add-author-book	addAuthor	作者名、作者年龄、作者国籍

3.5. 各 Action/前端页面之间的调用和消息传递关系



4. 图书 SaaS 核心代码

4.1. 按作者查询

JSP 代码（以下代码在 form 中）：

```
<div class="form-group">
  <label
    class="control-label col-md-3 col-sm-3 col-xs-12"
    for="authorName">作者姓名
  </label>
  <div class="col-md-6 col-sm-6 col-xs-12">
    <input type="text" id="authorName"
      name="authorName"
      class="form-control col-md-7 col-xs-12" />
  </div>
</div>
```

Action 类代码：

```
public class TableBookAction extends ActionSupport {
    public String publisher, publishDate, authorName, bookName;
    public String execute() {
        return SUCCESS;
    }
    public String getPublisher() {
        return publisher;
    }
    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }
    public String getPublishDate() {
        return publishDate;
    }
    public void setPublishDate(String publishDate) {
        this.publishDate = publishDate;
    }
    public String getAuthorName() {
        return authorName;
    }
    public void setAuthorName(String authorName) {
        this.authorName = authorName;
```

```
    }
    public String getBookName() {
        return bookName;
    }
    public void setBookName(String bookName) {
        this.bookName = bookName;
    }
}
```

Action 类只需要将用户的查询关键字传递给 tableBook.jsp 即可。tableBook.jsp 使用了 JS 来处理用户的查询关键字，以此来显示用户想要查询的书籍信息。

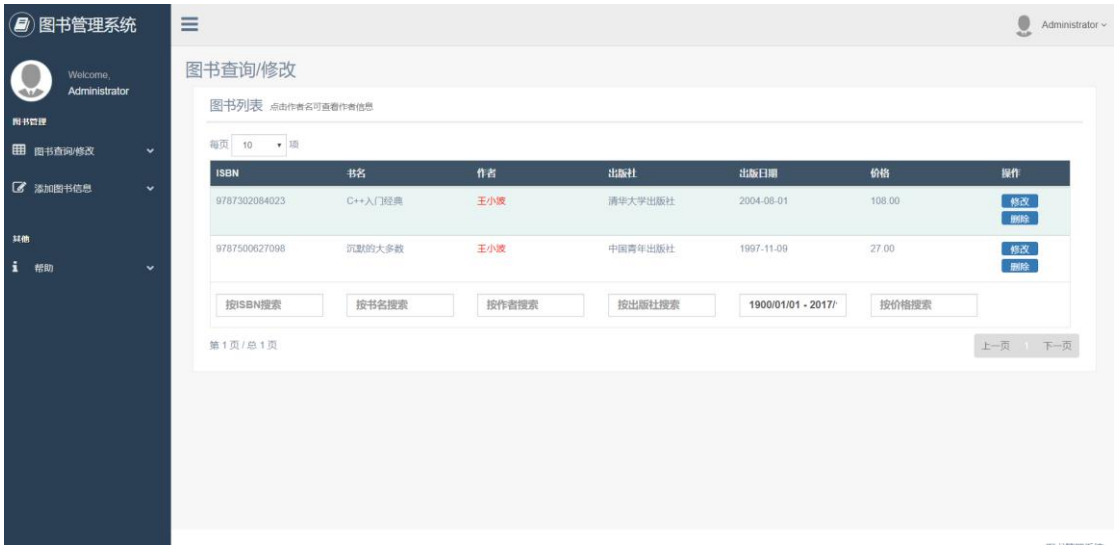
4.2. 展示图书详细信息

前端 JS 代码：

```
    var table = $("#tableBook").DataTable({
        "serverSide" : true,
        "ajax" : {
            "url" : "/library/dataTableBook.action",
            "type" : "POST",
            "data" : function(d) {
                return $.extend({}, d, {
                    "intention" : "query"
                });
            }
        },
        "ordering" : false,
        "columns" : [ {
            "data" : "isbn"
        }, {
            "data" : "title"
        }, {
            "data" : "author",
            "render" : function(data, type, row) {
                return '<a href="javascript:void(0)" data-toggle="modal"
data-target="#show-author-modal" style="color:red" id="button-author">' +
data["name"] + '</a>';
            }
        }, {
            "data" : "publisher"
        }, {
            "data" : "publishDate"
        }, {
```

```
            "data" : "price"
        }, {
            "data" : null
        } ],
        "columnDefs" : [ {
            "targets" : -1,
            "data" : null,
            "defaultContent" : '<button id="button-modify" type="button"
style="height:18px;width:50px;font-size:13px;vertical-align:middle;padding:
0;" \
                class="btn btn-primary" data-toggle="modal"
data-target="#modify-book-modal">修改</button>\
                <button id="button-delete" type="button"
style="height:18px;width:50px;font-size:13px;vertical-align:middle;padding:
0;" \
                class="btn btn-primary" data-toggle="modal"
data-target="#delete-book-modal">删除</button>'
        } ],
        'dom' : '<"float_left"f>r<"float_right"l>tip',
        'language' : {
            'emptyTable' : '没有数据',
            'loadingRecords' : '加载中...',
            'processing' : '查询中...',
            'search' : '搜索:',
            'lengthMenu' : '每页 _MENU_ 项',
            'zeroRecords' : '没有数据',
            'paginate' : {
                'first' : '第一页',
                'last' : '最后一页',
                'next' : '下一页',
                'previous' : '上一页'
            },
            'info' : '第 _PAGE_ 页 / 总 _PAGES_ 页',
            'infoEmpty' : '没有数据',
            'infoFiltered' : '(过滤总件数 _MAX_ 条)'
        }
    });
```

前端使用了 DataTable.js 插件，来进行图书信息的展示，因此在此放了 JS 代码。



后端需要处理 DataTable 的请求，返回查询结果（json 格式）。

```
        String preSql = "SELECT b.ISBN, b.Title, a.Name, a.Age, a.Country,
b.Publisher, b.PublishDate, b.Price "
                + "FROM Book b, Author a "
                + "WHERE a.AuthorID=b.AuthorID "
                + "and b.ISBN LIKE ? "
                + "and b.Title LIKE ? "
                + "and a.Name LIKE ? "
                + "and b.Publisher LIKE ? "
                + "and b.PublishDate >= ? "
                + "and b.PublishDate <= ? "
                + "and b.Price <= ? "
                + "order by b.ISBN "
                + "limit ?,?;";
        System.out.println(preSql);
        PreparedStatement preSta = connection.prepareStatement(preSql);
        ActionContext context=ActionContext.getContext();
        HttpServletRequest request =
(HttpServletRequest)context.get(ServletActionContext.HTTP_REQUEST);
        for (int i = 0; i < 6; i++) {
            if (i < 4)
                preSta.setString(i+1, "%"+request.getParameter("columns[" +
Integer.toString(i) + "][search][value]")+"%");
            else if (i == 4) {
                String dateRange =
request.getParameter("columns[4][search][value]");
                if (dateRange.equals("") || dateRange == null)
                    dateRange = "1900/01/01-2017/09/01";
                System.out.println(dateRange);
```

```
System.out.println(dateRange.split("-")[0].trim().replace('/', '-'));
                preSta.setString(i+1,
dateRange.split("-")[0].trim().replace('/', '-'));
                preSta.setString(i+2,
dateRange.split("-")[1].trim().replace('/', '-'));
                System.out.println(dateRange.split("-")[1].replace('/',
'-'));
            } else {
                String price = request.getParameter("columns[" +
Integer.toString(i) + "][search][value]");
                if (price.equals(""))
                    price = "10000000.0";
                preSta.setFloat(i+2, Float.parseFloat(price));
            }
        }
        preSta.setInt(8, Integer.parseInt(request.getParameter("start")));
        preSta.setInt(9, Integer.parseInt(request.getParameter("length")));
        System.out.println(preSta.toString());
        ResultSet resultSet = preSta.executeQuery();
        data = new ArrayList();
        recordsFiltered = 0;
        while (resultSet.next()){
            HashMap row = new HashMap();
            HashMap author = new HashMap();
            for (int i = 1; i <= 8; i++) {
                if (i >=3 && i <= 5)
                    author.put(col[i-1], resultSet.getString(i));
                else
                    row.put(col[i-1], resultSet.getString(i));
                System.out.println(resultSet.getString(i));
            }
            row.put("author", author);
            data.add(row);
            ++recordsFiltered;
        }

        preSta = connection.prepareStatement("select count(*) from Book");
        resultSet = preSta.executeQuery();
        while (resultSet.next()) {
            recordsTotal = Integer.parseInt(resultSet.getString(1));
        }
```

DataTableBookAction 类中，需要修改 data、recordsTotal 和 recordsFiltered。data

是一个 ArrayList，存储的是查询结果中的书籍集合。recordsTotal 是书籍的总数，recordsFiltered 是过滤出的书籍的数目。struts-json 会将这三者解析为 json 返回给前端。

4.3. 删除图书

前端 JS 代码：

```
$("#tableBook").on("click","#button-delete",function(){
    if(!confirm("确定删除?"))
        return;
    var ISBN = table.row($(this).parents("tr")).data()["isbn"];
    $.ajax({
        url: "/library/modifyBook.action",
        type: "POST",
        data: {
         "ISBN":ISBN,
         "intention" : "delete"
    },
        success: (function(data){
         if (data["ok"] == true) {
                var t = $("#tableBook").dataTable();
                t.fnReloadAjax();
                alert("删除成功");
         } else {
             alert("删除失败");
         }
        }),
        error: (function(data){
            alert("删除失败");
        })
    });
});
```

Action 代码：

```
preSql = "DELETE FROM Book WHERE ISBN = ?";
System.out.println(preSql);
PreparedStatement preSta = connection.prepareStatement(preSql);
preSta.setString(1, ISBN);
result = preSta.executeUpdate();
```

4.4. 新增图书/作者(可选)

新增图书的前端 JS 代码：

```
function submit_modify_book_form() {
    var dataPara = getFormJson($("#add-book-form"));
    dataPara["intention"] = "add";
    $.ajax({
        url : "/library/modifyBook.action",
        type : "POST",
        data : dataPara,
        success : function(data) {
            if (data["ok"] == true) {
                alert("添加成功！");
            } else {
                alert("添加失败，请重新检查输入！");
            }
        },
        error : function() {
            alert("添加失败，请重新检查输入！");
        }
    });
}
function getFormJson(node) {
    var o = {};
    node.find("input,select").each(function(){
        if (o[this.name] !== undefined) {
            if (!o[this.name].push) {
                o[this.name] = [o[this.name]];
            }
            o[this.name].push(this.value || '');
        } else {
            o[this.name] = this.value || '';
        }
    });
    return o;
}
```

新增图书的后端 Action 代码：

```
            preSql = "insert into Book (ISBN, Title, AuthorID, Publisher,
PublishDate, Price) values (?, ?, ?, ?, ?, ?);";
            System.out.println(preSql);
            PreparedStatement preSta = connection.prepareStatement(preSql);
```

```
            preSta.setString(1, ISBN);
            if (!intention.equals("delete")) {
                preSta.setString(2, title);
                preSta.setInt(3, authorID);
                preSta.setString(4, publisher);
                preSta.setString(5, publishDate);
                preSta.setFloat(6, price);
            }
            System.out.println(preSta.toString());
            result = preSta.executeUpdate();
```

新增作者的前端 JS 代码：

```
function submit_add_author_form() {
    var dataPara = getFormJson($("#add-author-form"));
    $.ajax({
        url : "/library/addAuthor.action",
        type : "POST",
        data : dataPara,
        success : function(data) {
            if (data["ok"] == false) {
                alert("添加失败，请重新检查输入！");
            } else {
                alert("添加成功！");
            }
        },
        error : function() {
            alert("添加失败，请重新检查输入！");
        }
    });
}
```

新增作者的后端 Action 代码：

```
            String preSql = "insert into Author (Name, Age, Country) values
(?, ?, ?);";

            System.out.println(preSql);
            PreparedStatement preSta = connection.prepareStatement(preSql);
            preSta.setString(1, name);
            preSta.setInt(2, age);
            preSta.setString(3, country);
            System.out.println(preSta.toString());
            result = preSta.executeUpdate();
```

4.5. 更新图书信息(可选)

前端 JSP 代码：

```
        <form action="/library/tableBook.action" method="post"
    enctype="multipart/form-data"
            class="form-horizontal form-label-left" data-toggle="validator"
role="form" id="modify-book-form">

            <div class="form-group">
              <label
                class="control-label col-md-3 col-sm-3 col-xs-12"
                for="ISBN">ISBN
              </label>
              <div class="col-md-6 col-sm-6 col-xs-12">
                <div class="input-group" style="margin:0;padding:0">
                  <input type="text" id="isbn"
                    name="ISBN"
                    class="form-control" required  disabled="disabled" />
                  <span class="input-group-btn">
                    <button type="button" class="btn btn-primary"
style="margin-right:0" id="get-douban-info">使用豆瓣API获取书籍数据</button>
                  </span>
                </div>
              </div>
              <div class="help-block with-errors"></div>
            </div>
            <div class="form-group">
              <label
                class="control-label col-md-3 col-sm-3 col-xs-12"
                for="title">书名
              </label>
              <div class="col-md-6 col-sm-6 col-xs-12">
                <input type="text" id="title"
                  name="title"
                  class="form-control col-md-7 col-xs-12" required/>
              </div>
              <div class="help-block with-errors"></div>
            </div>
            <div class="form-group">
              <label for="publisher"
                class="control-label col-md-3 col-sm-3 col-xs-12">作者
```

```
</label>
              <div class="col-md-6 col-sm-6 col-xs-12">
                <div class="input-group"
style="margin:0;padding:0;width:100%">
                  <select class="js-states form-control" name="authorID"
id="authorID" style="width:100%" required></select>
                </div>
              </div>
              <div class="help-block with-errors"></div>
            </div>
            <div class="form-group">
              <label for="publisher"
                class="control-label col-md-3 col-sm-3 col-xs-12">出版社
</label>
              <div class="col-md-6 col-sm-6 col-xs-12">
                <input type="text" id="publisher"
                  class="form-control col-md-7 col-xs-12"
                  name="publisher" required/>
              </div>
              <div class="help-block with-errors"></div>
            </div>
            <div class="form-group">
              <label for="publishDate"
                class="control-label col-md-3 col-sm-3 col-xs-12">出版日期
              </label>
              <div class="col-md-6 col-sm-6 col-xs-12">
                <input type="text" id="publishDate" name="publishDate"
                  class="date-picker form-control col-md-7 col-xs-12"
pattern="[0-9]+-[0-9]+-[0-9]+"
                  type="text" required/>
              </div>
              <div class="help-block with-errors"></div>
            </div>
            <div class="form-group">
              <label for="price"
                class="control-label col-md-3 col-sm-3 col-xs-12">价格
              </label>
              <div class="col-md-6 col-sm-6 col-xs-12">
                <div class="input-group" style="margin:0;padding:0">
                  <span class="input-group-addon">￥</span>
                  <input type="number" id="price"
                    name="price"
                    class="form-control col-md-7 col-xs-12" placeholder="e.g.
10.00" required />
```

```
                </div>
              </div>
              <div class="help-block with-errors"></div>
            </div>
<!--              <div class="ln_solid"></div> -->
            <div class="form-group">
              <div
                class="col-md-6 col-sm-6 col-xs-12 col-md-offset-3"
style="text-align:center">
                <button class="btn btn-primary" type="reset"
style="display:none">重置</button>
                <button type="submit" class="btn btn-success"
style="display:none">提交</button>
              </div>
            </div>

          </form>
```

前端 JS 代码：

```
function submit_modify_book_form() {
    var dataPara = getFormJson($("#modify-book-form"));
    dataPara["intention"] = "modify";
    $.ajax({
        url : "/library/modifyBook.action",
        type : "POST",
        data : dataPara,
        success : function(data) {
            if (data["ok"] == true) {
                var t = $("#tableBook").dataTable();
                t.fnReloadAjax();
                $("#modify-book-modal").modal("hide");
            } else {
                alert("修改失败，请重新检查输入！");
            }
        },
        error : function() {
            alert("修改失败，请重新检查输入！");
        }
    });
}
```

后端 Action 代码：

```
            String preSql = "replace into Book set ISBN=?,
Title=?,AuthorID=?,Publisher=?,PublishDate=?, Price=?;";
            System.out.println(preSql);
            PreparedStatement preSta = connection.prepareStatement(preSql);
            preSta.setString(1, ISBN);
            if (!intention.equals("delete")) {
                preSta.setString(2, title);
                preSta.setInt(3, authorID);
                preSta.setString(4, publisher);
                preSta.setString(5, publishDate);
                preSta.setFloat(6, price);
            }
            System.out.println(preSta.toString());
            result = preSta.executeUpdate();
```

4.6. 数据库连接与访问

```
        Class.forName("com.mysql.jdbc.Driver");
        String url =
"jdbc:mysql://localhost:3306/BookDB?useUnicode=true&useJDBCCompliantTimezon
eShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";//忽略

        Connection connection = DriverManager.getConnection(url, "root",
"AdHoc987");        //关闭语句
        Statement sta = connection.createStatement();
        sta.close();
        //关闭结果集
        resultSet.close();
        //关闭数据库连接
        connection.close();
```

5. 图书 SaaS 的云平台部署

5.1. 所选定的公共云平台

本人选择了阿里云 IaaS 的 ecs.n4.small 服务器，1 核 2GB 内存，地区华北二 E 区。

5.2. 部署步骤

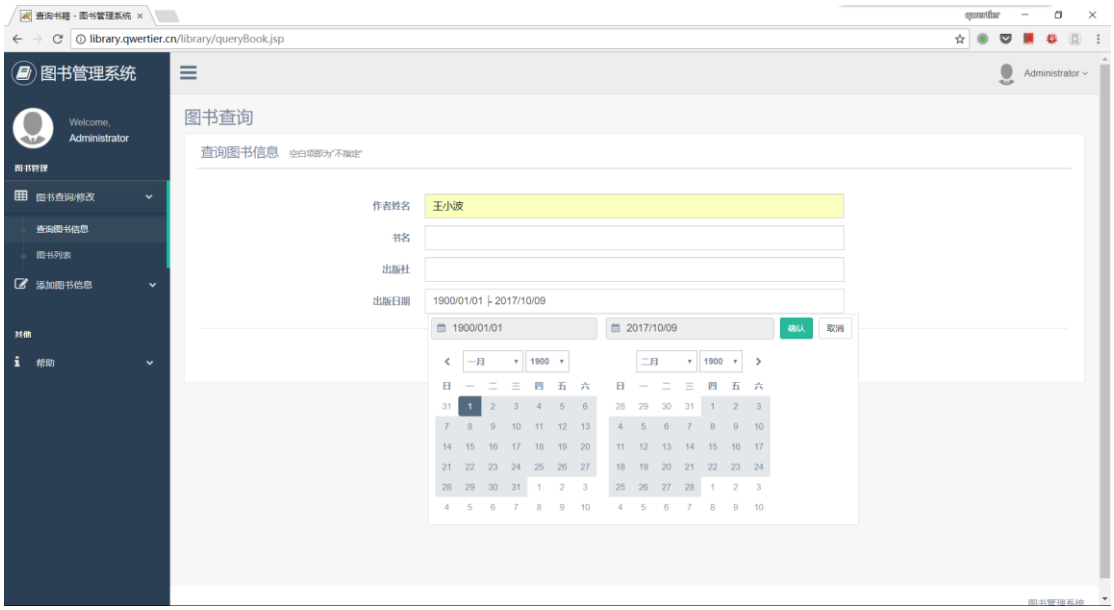
实际上，在阿里云服务器上部署网站的难度与在本地部署相近（由于 apt-get 的易用性，在 linux 服务器上部署甚至更简单）。

我先是安装了 oracle-java-8，设置完 JAVA_HOME 和 JAVA_JRE 环境变量之后，便安装 MySQL，在安装 MySQL 的过程中便完成了用户名和密码的设置（比 Windows 好用多了）。

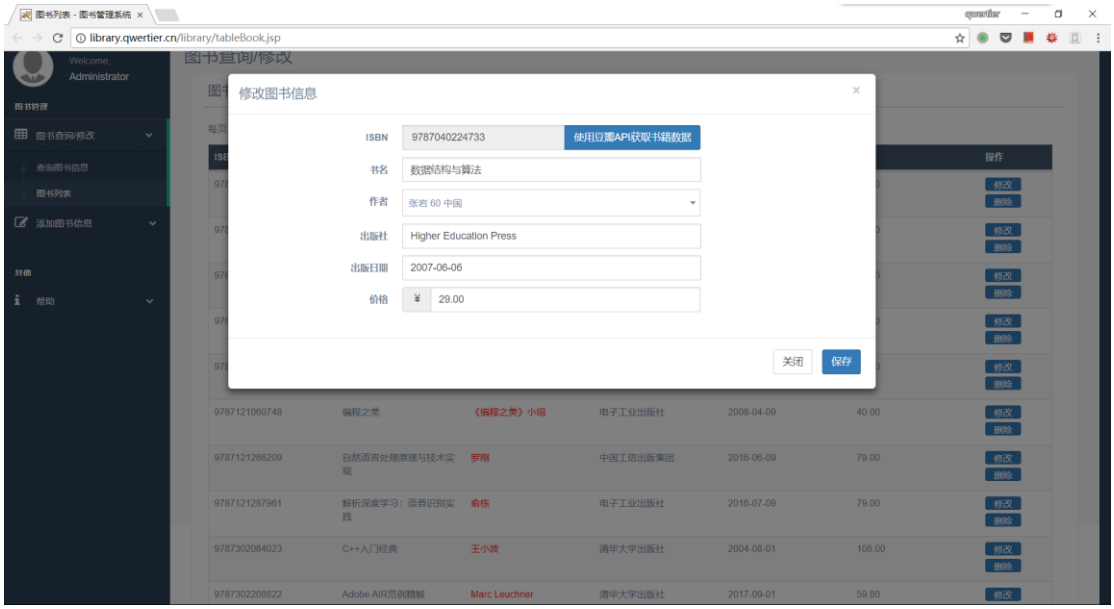
之后，我安装了 Tomcat 9，用 systemctl 将 Tomcat 9 设为服务之后，便启动了 Tomcat，但这时，却无法在外网访问 Tomcat 的默认页面，我后来才发现自己没有在服务器的安全选项中打开 8080 端口的外部访问权限，后来我又打开了 80 端口的权限，Tomcat 顿时就可以在外网访问了。

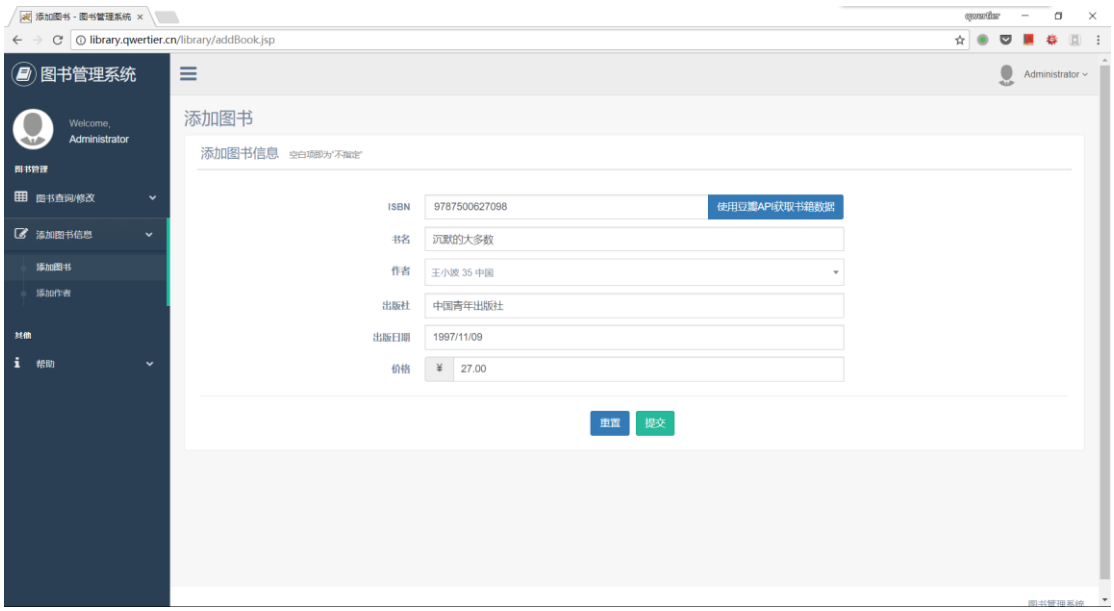
部署完网站环境之后，我把 war 文件打包，用 scp 上传到服务器上，并重启 Tomcat 服务，并使用 iptables 将 80 端口映射到 8080 端口，再将 library.qwertier.cn 解析到阿里云服务器上，顿时便可以通过 http://library.qwertier.cn/library 访问图书 SaaS 了。

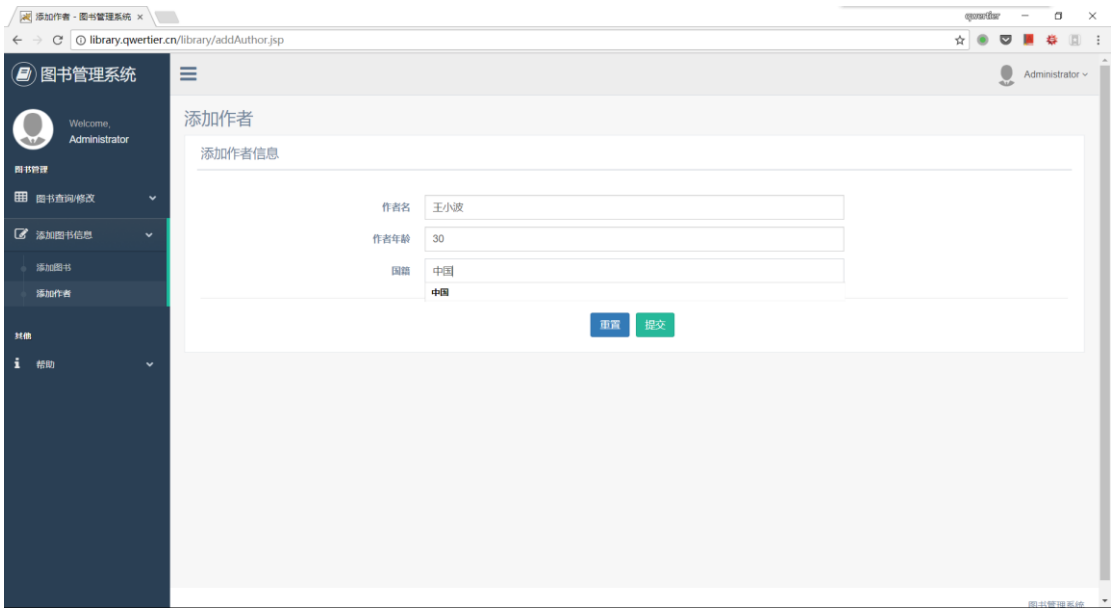
5.3. 外在访问结果




library.qwertier.cn/library/tableBook.jsp
Welcome,
Administrator
图书查询/修改
图书列表
ISBN
书名
作者
出版社
出版日期
价格
操作


library.qwertier.cn/library/tableBook.jsp
修改图书信息
ISBN
9787040224733
使用豆瓣API获取书籍数据
书名
数据结构与算法
作者
张岩 60 中国
出版社
Higher Education Press
出版日期
2007-06-06
价格
¥
29.00
关闭
保存


library.qwertier.cn/library/addBook.jsp
图书管理系统
Welcome,
Administrator
添加图书
添加图书信息
ISBN
9787500627098
使用豆瓣API获取书籍数据
书名
沉默的大多数
作者
王小波 35 中国
出版社
中国青年出版社
出版日期
1997/11/09
价格
¥
27.00
重置
提交



6. 小结

- 与在本地环境部署 Web 应用相比，在云平台上部署 Web 应用的难度体现在？你是如何克服这些难点的？
  - 在云平台上部署的难点体现在 Linux 操作系统的使用上。实际上，由于长期使用 Linux 的经历，在 Linux 上进行部署我个人认为更为简单（特别是安装步骤）。
- 与之前你擅长的桌面程序开发相比，Web 开发有什么不一样的地方？
  - 使用 Java 进行 Web 开发，与实验一中使用 Java 的 Swing 进行开发，最大的不一样的地方在于：设计与逻辑分开编写。我个人认为 Web 开发更适合我。
- 面对全新的技术，你如何做到快速从基础为 0 到能够用其开发一个简单的原型 Web 系统？
  - 大量查阅文档、教程网站（例如 w3school）。多阅读，多思考。
- 你是否适应了在短时间内“目标驱动（开发一个具体的系统）”来学习一种新技术？
  - 是的，自从大一的 Python 课程开始，我便开始学着使用目标来驱动自己学习技术，来完成项目（大部分为作业）。
- 过去两周，你对 Lab2 贡献了多少精力和时间？值得吗？如果想骂人，可以在这里骂出来~~ 但如果你觉得有收获，不妨也在这里简单记录下来。
  - 贡献了至少有 5 天的时间，虽然实际上大部分代码都是在最后两天写的。首先 struts 这个框架我认为自己已经有了简单的认识，并且可以将来独立完成一个较为简单的网站，我想这是我最大的收获。
- 为了在云端调试这个简单的 Web 系统，你花了多少钱？
  - 10 元。