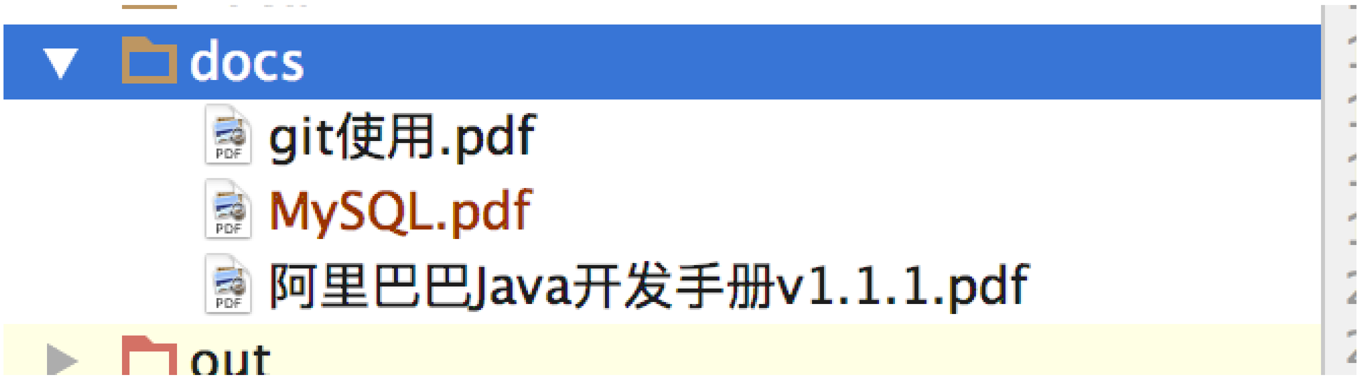
梳理一下项目各级目录关系

docs

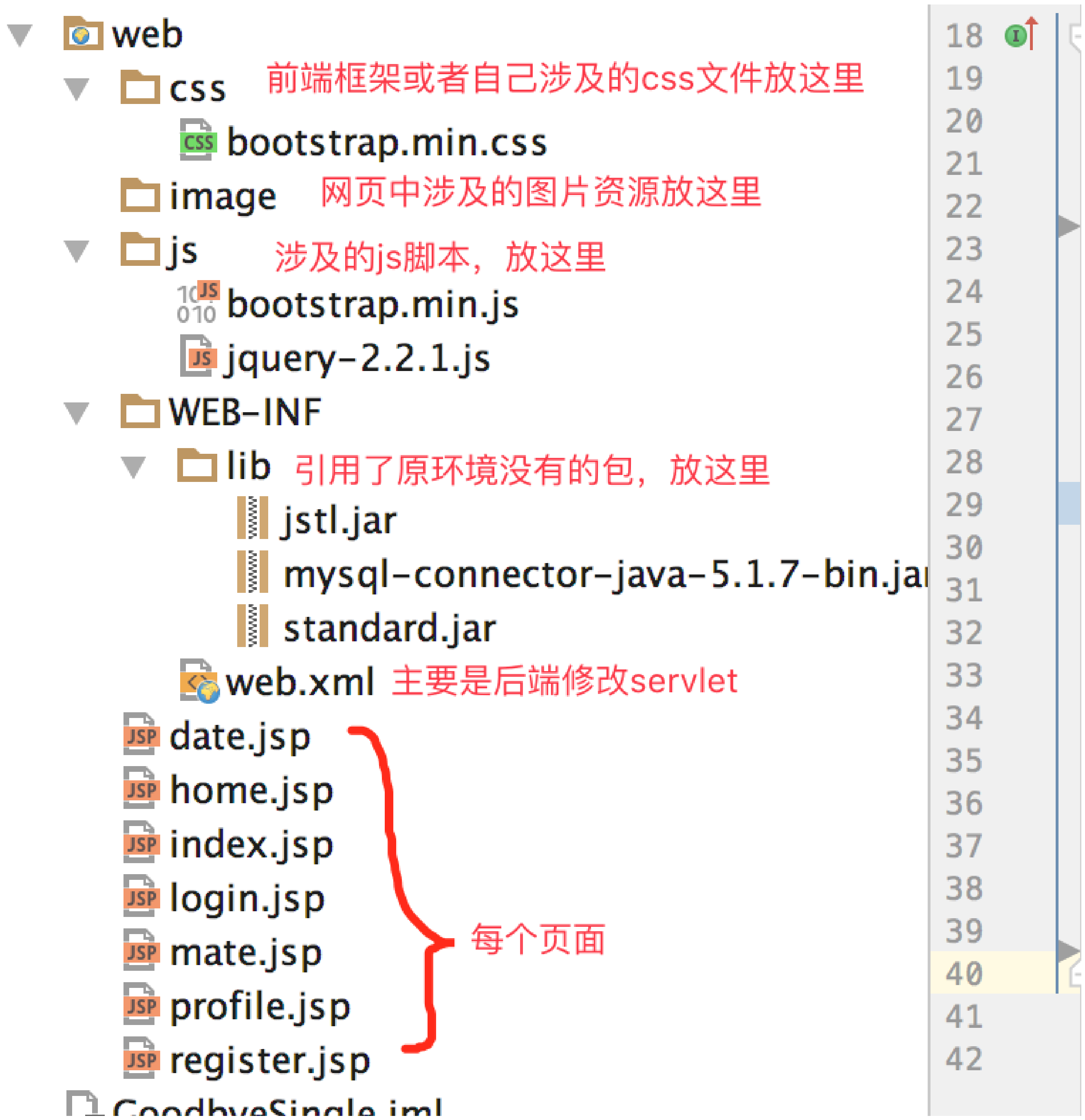
这个目录下放了一些帮助文档，如果你敲代码的时候某个部分不清晰，可以看看帮助文档。



docs
git使用.pdf
MySQL.pdf
阿里巴巴Java开发手册v1.1.1.pdf
out

web(前端)

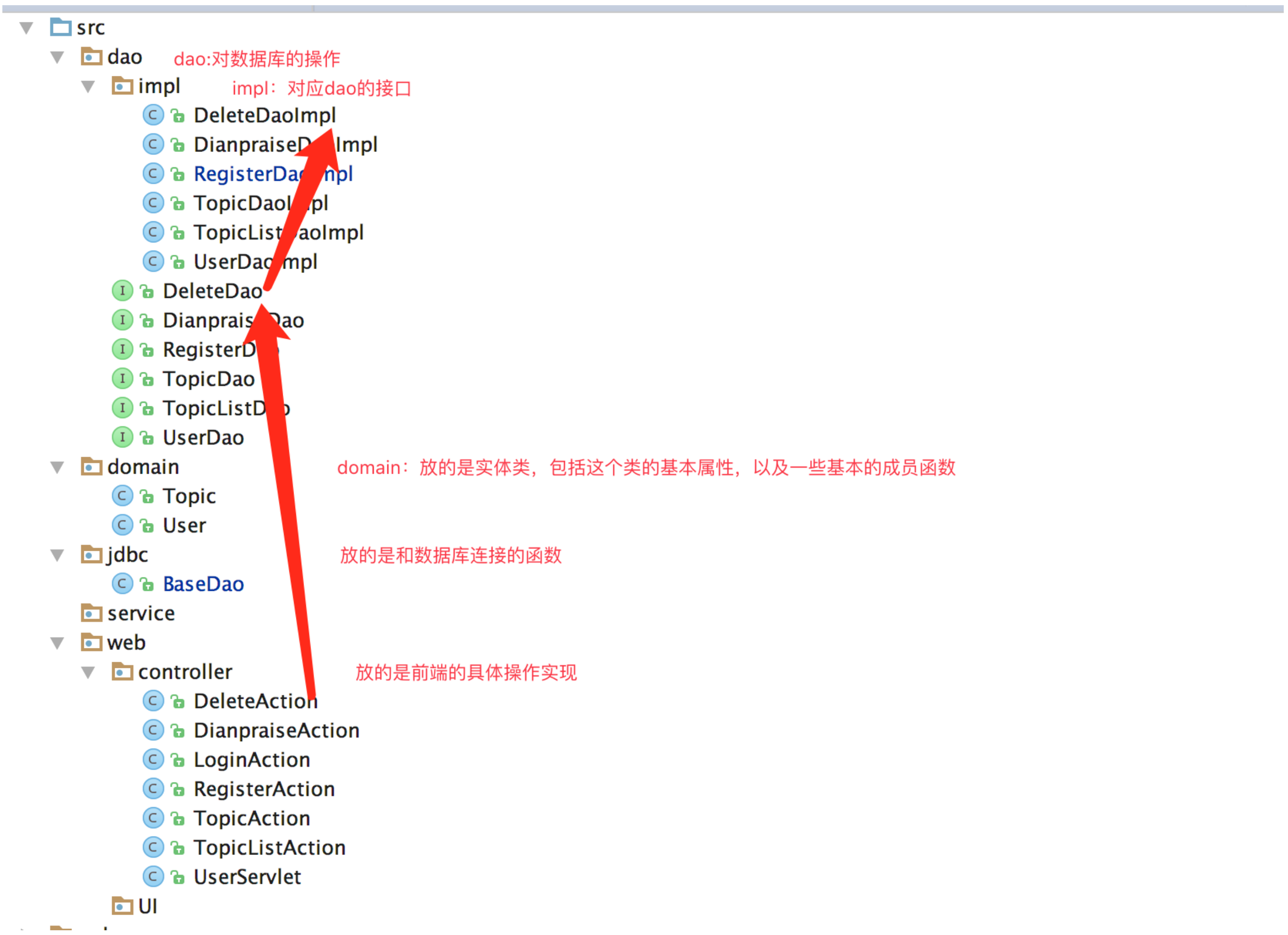
这个文件夹下面放的是前端的css、js以及jsp文件。



```
web
  css        前端框架或者自己涉及的css文件放这里
    bootstrap.min.css
  image      网页中涉及的图片资源放这里
  js         涉及的js脚本，放这里
    bootstrap.min.js
    jquery-2.2.1.js
  WEB-INF
    lib      引用了原环境没有的包，放这里
      jstl.jar
      mysql-connector-java-5.1.7-bin.jar
      standard.jar
    web.xml  主要是后端修改servlet
  date.jsp
  home.jsp
  index.jsp
  login.jsp      每个页面
  mate.jsp
  profile.jsp
  register.jsp
GoodbyeSingle.iml
```

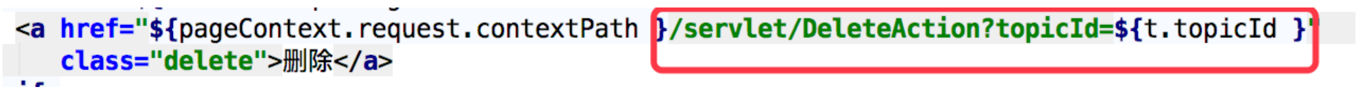
src（后端）

这个目录下放的是后端的业务逻辑实现代码

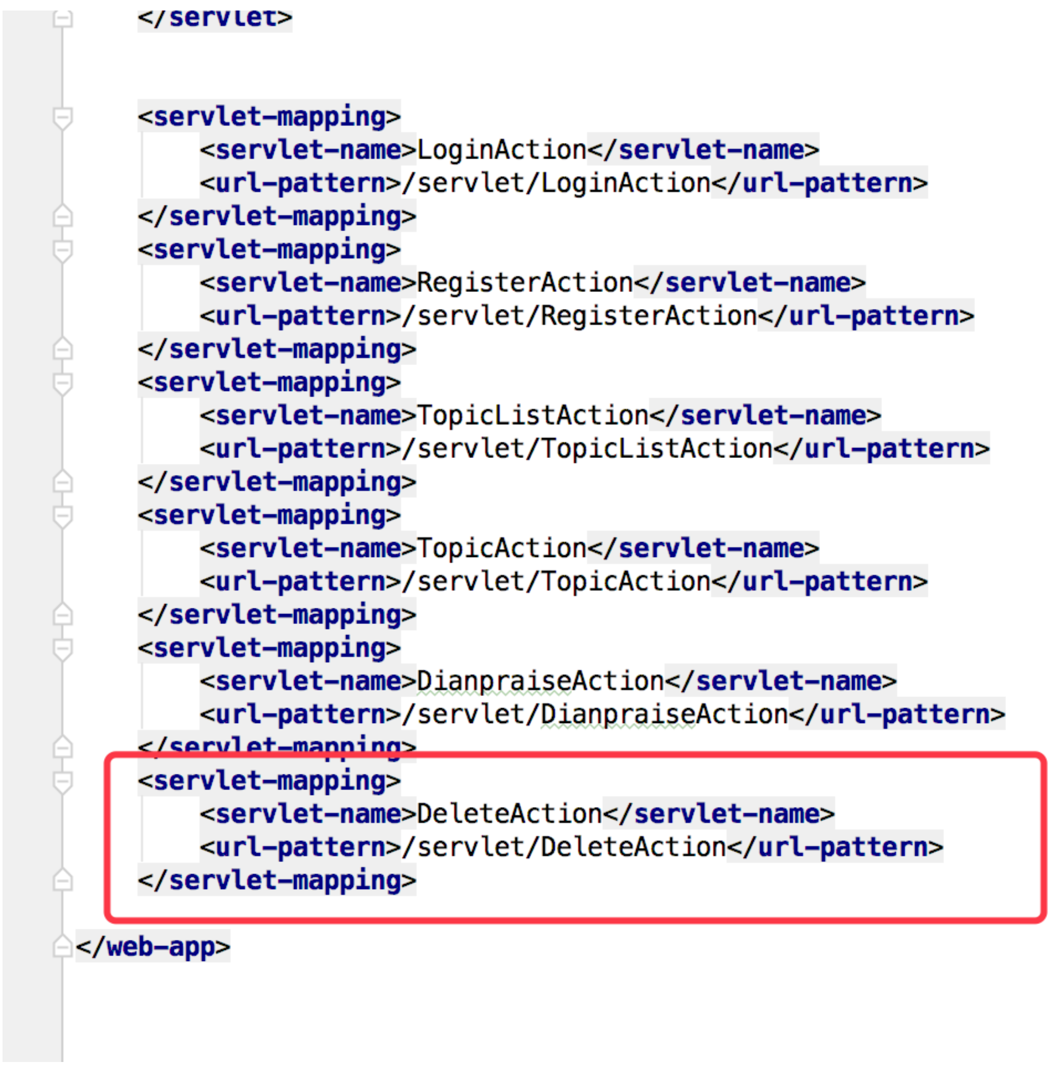

```
src
  dao          dao:对数据库的操作
    impl         impl：对应dao的接口
      DeleteDaoImpl
      DianpraiseDaoImpl
      RegisterDaoImpl
      TopicDaoImpl
      TopicListDaoImpl
      UserDaoImpl
    DeleteDao
    DianpraiseDao
    RegisterDao
    TopicDao
    TopicListDao
    UserDao
  domain       domain：放的是实体类，包括这个类的基本属性，以及一些基本的成员函数
    Topic
    User
  jdbc         放的是和数据库连接的函数
    BaseDao
  service
  web
    controller   放的是前端的具体操作实现
      DeleteAction
      DianpraiseAction
      LoginAction
      RegisterAction
      TopicAction
      TopicListAction
      UserServlet
    UI
```

我们以一个例子来说明这个文件夹的关系。我们来看看前端点一个删除按钮的整个实现过程。

1.一个href为/servlet/DeleteAction的超链接

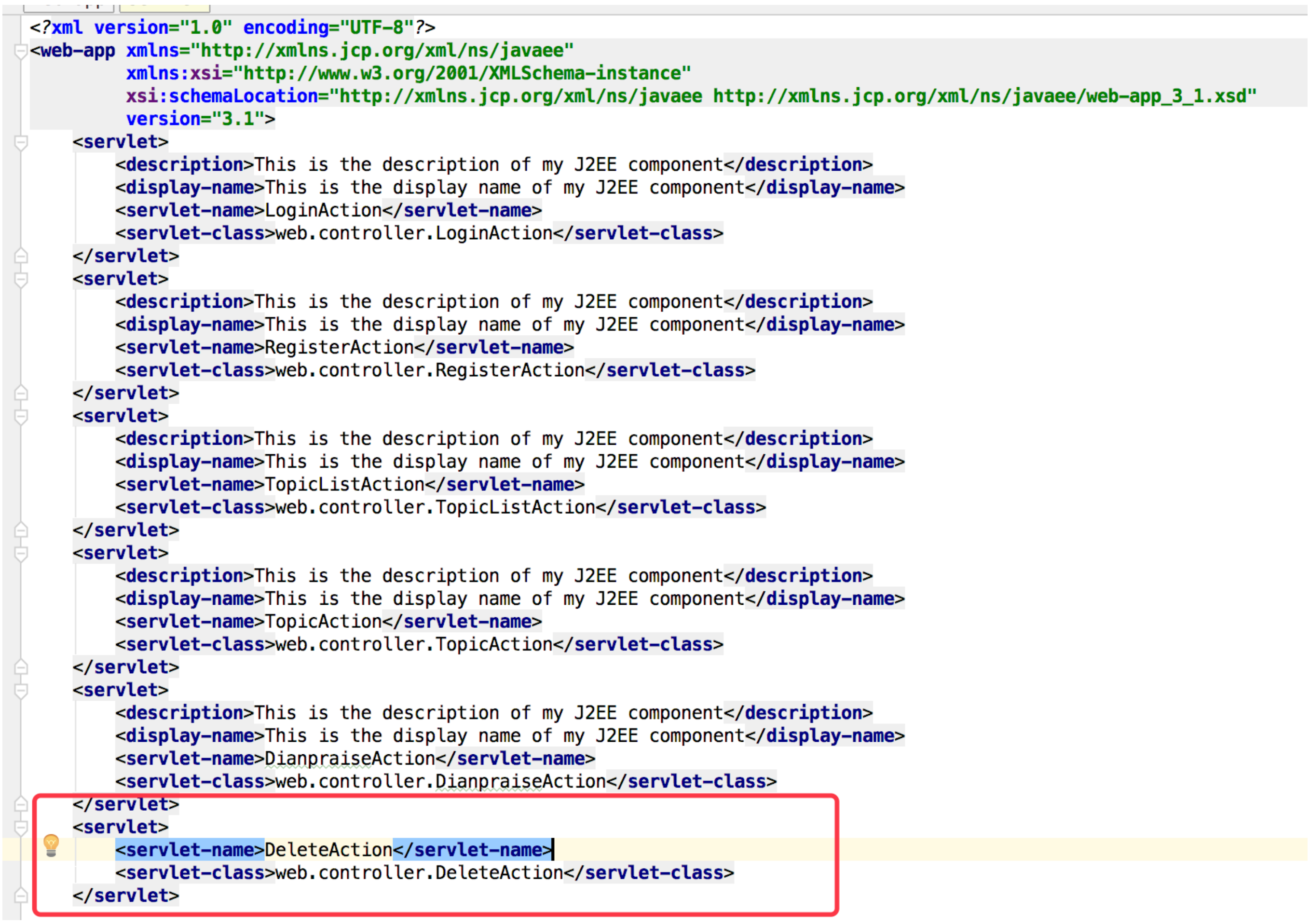

```
<a href="${pageContext.request.contextPath }/servlet/DeleteAction?topicId=${t.topicId }"
   class="delete">删除</a>
```

2.点击之后，它会去web/WEB-INF/web.xml文件中的servlet-mapping下找/servlet/DeleteAction，然后我们就知道了这个servlet的名字叫做DeleteAction



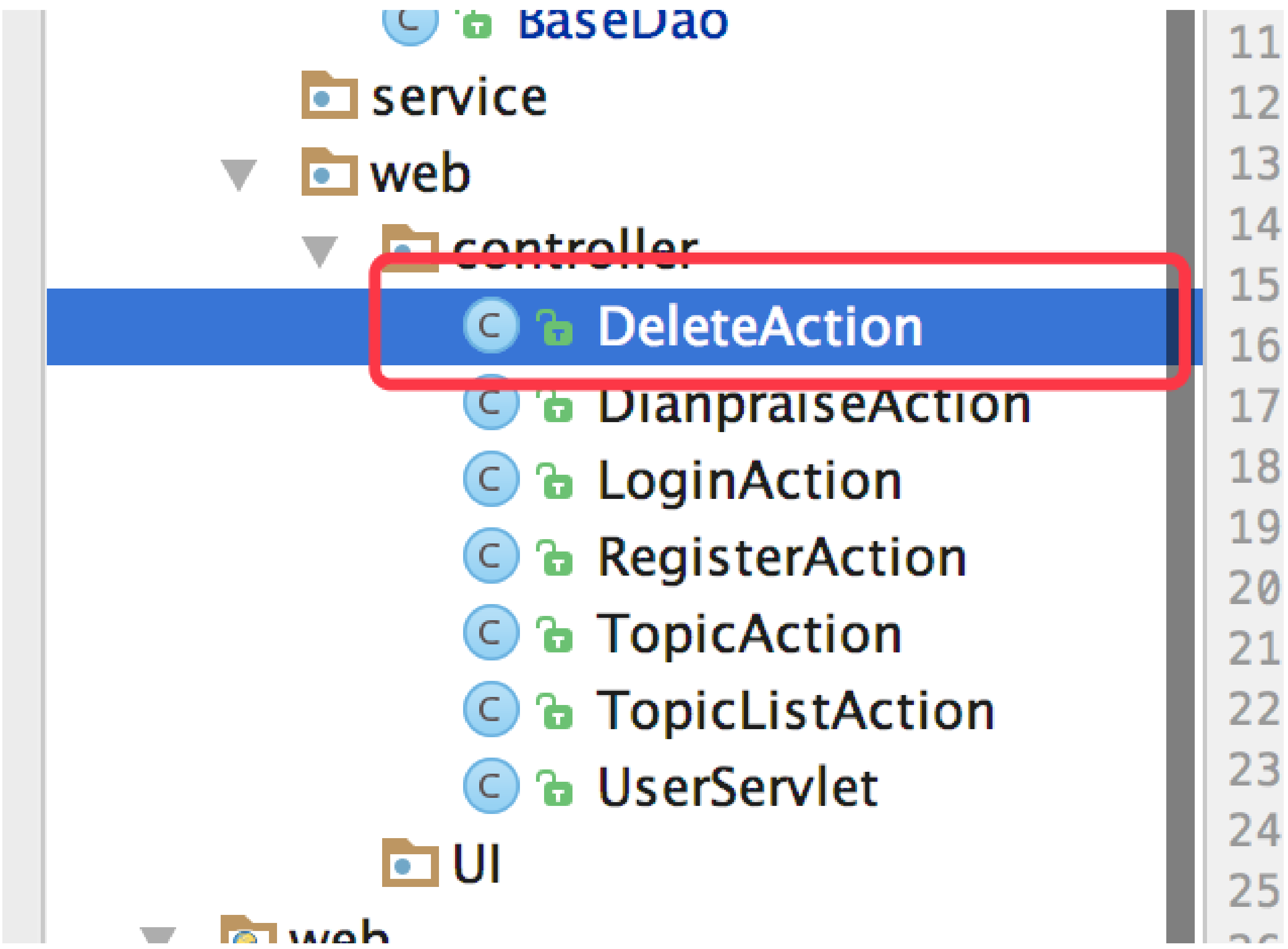
```
    </servlet>


    <servlet-mapping>
        <servlet-name>LoginAction</servlet-name>
        <url-pattern>/servlet/LoginAction</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>RegisterAction</servlet-name>
        <url-pattern>/servlet/RegisterAction</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>TopicListAction</servlet-name>
        <url-pattern>/servlet/TopicListAction</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>TopicAction</servlet-name>
        <url-pattern>/servlet/TopicAction</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>DianpraiseAction</servlet-name>
        <url-pattern>/servlet/DianpraiseAction</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>DeleteAction</servlet-name>
        <url-pattern>/servlet/DeleteAction</url-pattern>
    </servlet-mapping>

</web-app>
```

3.知道这个servlet的名字之后，会去上面servlet信息那儿找这个叫做DeleteAction的servlet，然后就找到了它的类，原来是在web.controller.DeleteAction

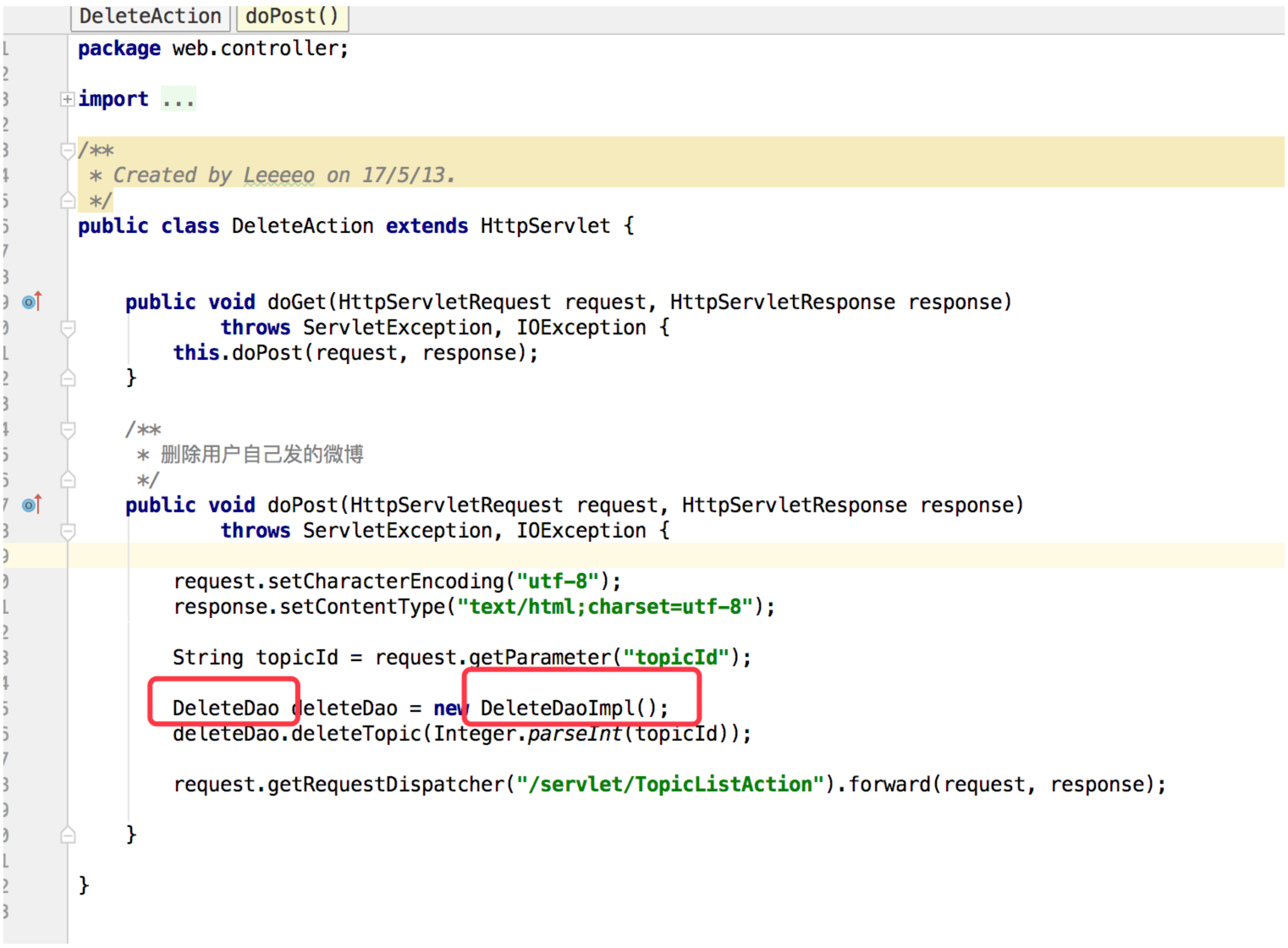


```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
    <servlet>
        <description>This is the description of my J2EE component</description>
        <display-name>This is the display name of my J2EE component</display-name>
        <servlet-name>LoginAction</servlet-name>
        <servlet-class>web.controller.LoginAction</servlet-class>
    </servlet>
    <servlet>
        <description>This is the description of my J2EE component</description>
        <display-name>This is the display name of my J2EE component</display-name>
        <servlet-name>RegisterAction</servlet-name>
        <servlet-class>web.controller.RegisterAction</servlet-class>
    </servlet>
    <servlet>
        <description>This is the description of my J2EE component</description>
        <display-name>This is the display name of my J2EE component</display-name>
        <servlet-name>TopicListAction</servlet-name>
        <servlet-class>web.controller.TopicListAction</servlet-class>
    </servlet>
    <servlet>
        <description>This is the description of my J2EE component</description>
        <display-name>This is the display name of my J2EE component</display-name>
        <servlet-name>TopicAction</servlet-name>
        <servlet-class>web.controller.TopicAction</servlet-class>
    </servlet>
    <servlet>
        <description>This is the description of my J2EE component</description>
        <display-name>This is the display name of my J2EE component</display-name>
        <servlet-name>DianpraiseAction</servlet-name>
        <servlet-class>web.controller.DianpraiseAction</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>DeleteAction</servlet-name>
        <servlet-class>web.controller.DeleteAction</servlet-class>
    </servlet>
```

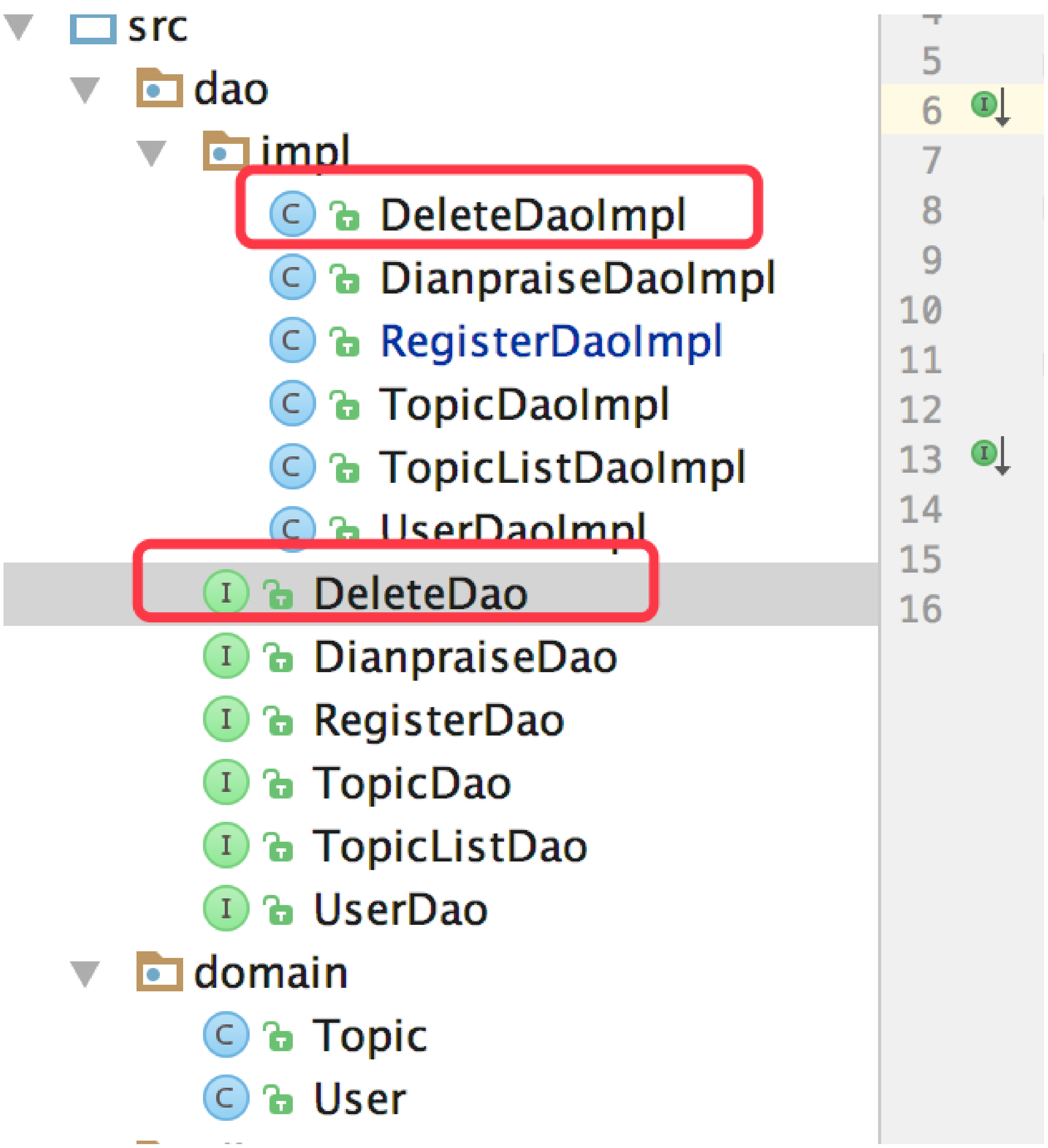
4.那么我们去到这个函数


BaseDao
service
web
controller
DeleteAction
DianpraiseAction
LoginAction
RegisterAction
TopicAction
TopicListAction
UserServlet
UI
web

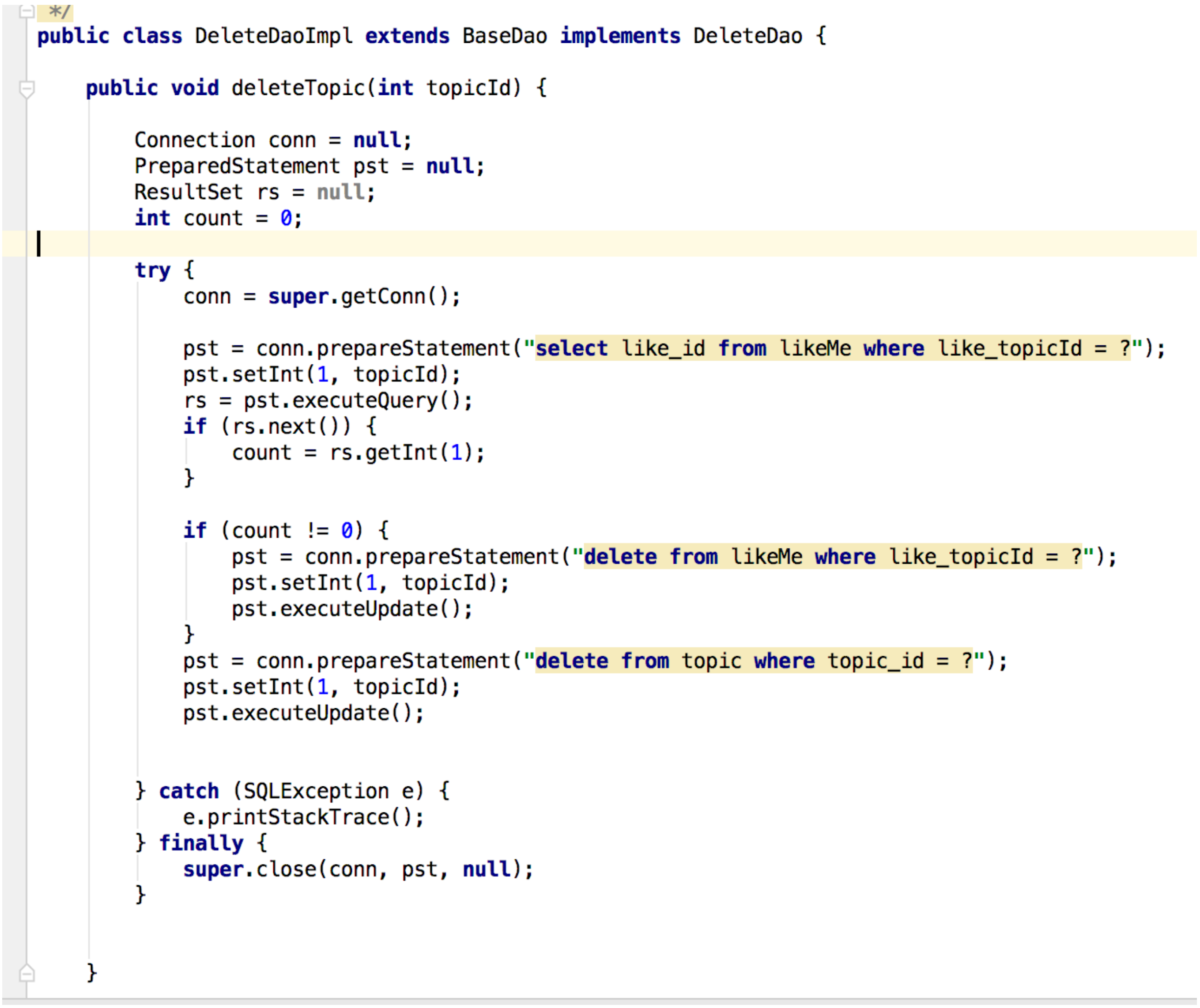
5.从上往下走代码，发现它又用到了DeleteDao 和 DeleteDaoImpl，所以我们去dao目录下找



```java
package web.controller;

import ...

/**
 * Created by Leeeeo on 17/5/13.
 */
public class DeleteAction extends HttpServlet {


    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        this.doPost(request, response);
    }

    /**
     * 删除用户自己发的微博
     */
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");

        String topicId = request.getParameter("topicId");

        DeleteDao deleteDao = new DeleteDaoImpl();
        deleteDao.deleteTopic(Integer.parseInt(topicId));

        request.getRequestDispatcher("/servlet/TopicListAction").forward(request, response);

    }

}
```

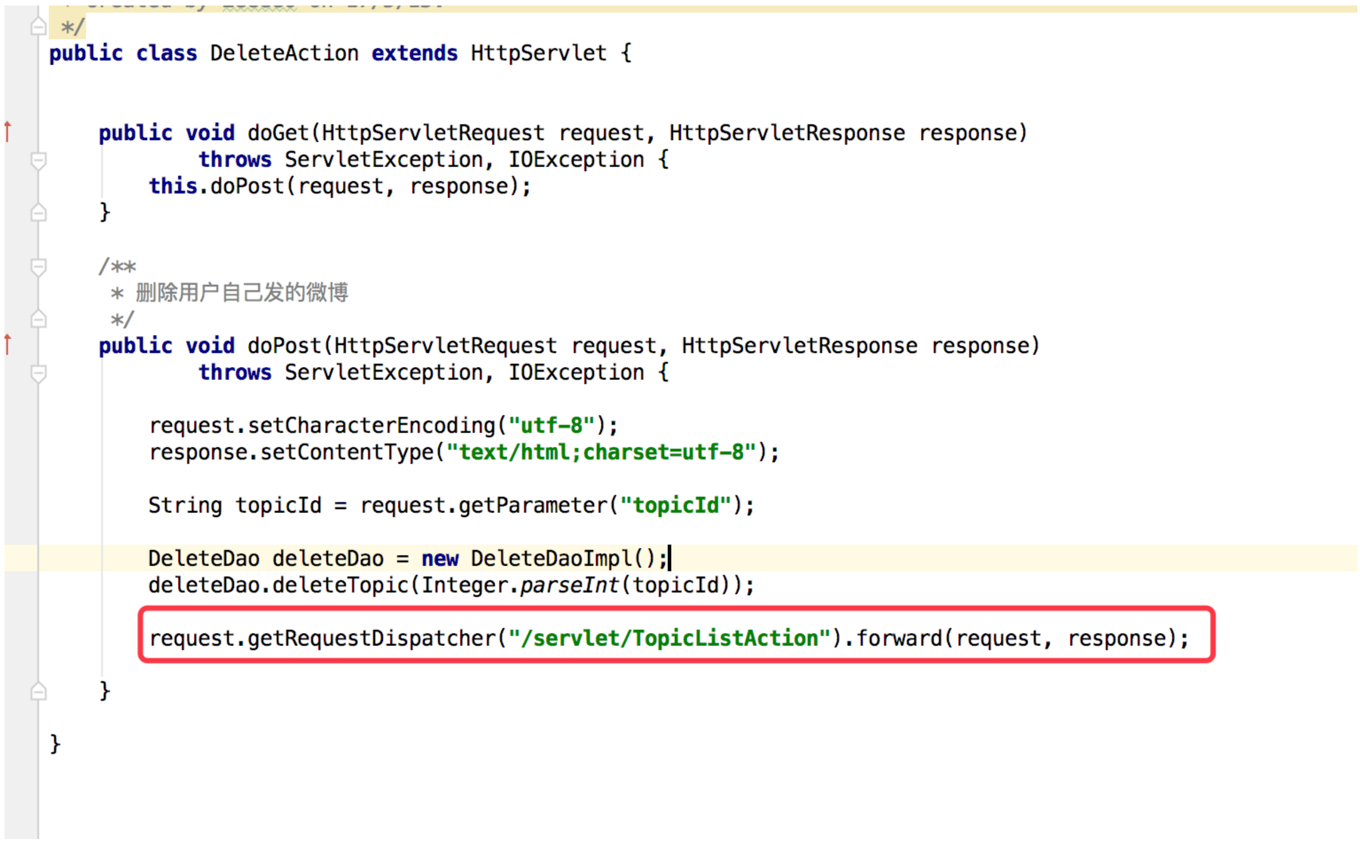


6.在DeleteDaoImpl接口文件中，我们看到了实现这个删除操作的具体代码



```
 */
public class DeleteDaoImpl extends BaseDao implements DeleteDao {

    public void deleteTopic(int topicId) {

        Connection conn = null;
        PreparedStatement pst = null;
        ResultSet rs = null;
        int count = 0;

        try {
            conn = super.getConn();

            pst = conn.prepareStatement("select like_id from likeMe where like_topicId = ?");
            pst.setInt(1, topicId);
            rs = pst.executeQuery();
            if (rs.next()) {
                count = rs.getInt(1);
            }

            if (count != 0) {
                pst = conn.prepareStatement("delete from likeMe where like_topicId = ?");
                pst.setInt(1, topicId);
                pst.executeUpdate();
            }
            pst = conn.prepareStatement("delete from topic where topic_id = ?");
            pst.setInt(1, topicId);
            pst.executeUpdate();


        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            super.close(conn, pst, null);
        }


    }
```

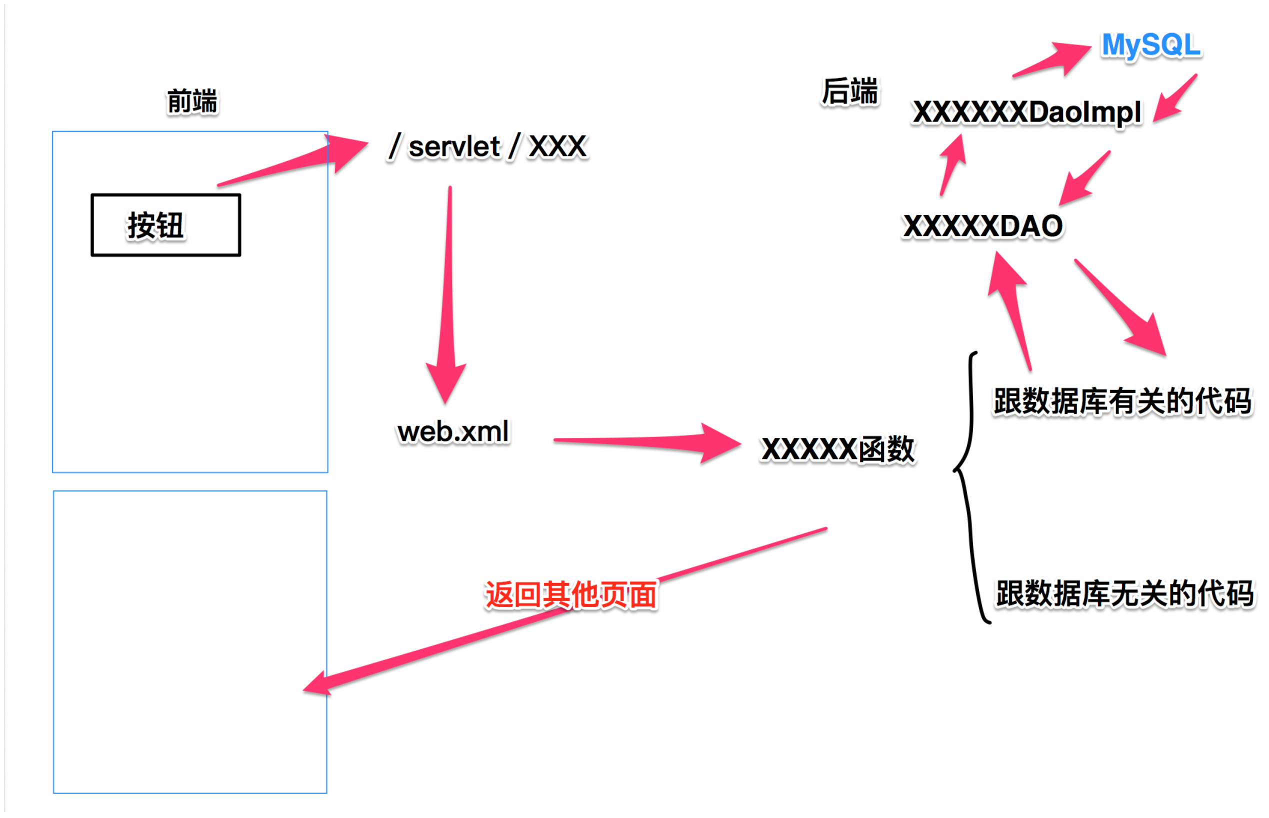
7.最后回到DeleteAction，发现它执行完了以后返回了/servlet/TopicListAction这个地址，至于后续如何就不说了，和前面流程一样，先去web.xml里找....



```
 */
public class DeleteAction extends HttpServlet {


    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        this.doPost(request, response);
    }

    /**
     * 删除用户自己发的微博
     */
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");

        String topicId = request.getParameter("topicId");

        DeleteDao deleteDao = new DeleteDaoImpl();
        deleteDao.deleteTopic(Integer.parseInt(topicId));

        request.getRequestDispatcher("/servlet/TopicListAction").forward(request, response);

    }

}
```

整体流程就是这样，总结一下

- web.xml:放servlet的name、class、mapping信息，用于前端匹配
- controller目录下放实现功能的代码
- dao目录下放与数据库操作有关的代码，调用impl下的具体实现



所以，当你编写代码之后发现不能工作，你可以按照上图的路径去检查，你是不是哪个地方漏掉了什么。