这段代码实现了一个CommentDao类，用于与数据库进行评论相关的操作。以下是对代码的分析和讲解：

1. 导入必要的包和类：

```java

import com.bean.Comment;

import com.util.DatabaseUtils;

```

这段代码导入了`Comment`类和`DatabaseUtils`工具类，用于操作评论实体和数据库连接。

总体来说，**CommentDao**类用于封装评论相关的数据库操作，包括添加评论、点赞评论、删除评论和检查评论是否属于指定用户。这些方法通过使用**PreparedStatement**对象和数据库连接，执行相应的SQL语句来实现相应的功能，并在必要时进行异常处理。

2. 构造方法和字段：

这段代码中的`CommentDao`类没有显示的定义构造函数或字段。

3. `likeComment(int commentId)`方法：

这个方法用于增加评论的点赞数。它的参数是commentId（评论ID），返回点赞数。

- 该方法首先获取数据库连接，然后使用预处理语句查询指定commentId的当前点赞数。

- 然后使用另一个预处理语句更新点赞数，并返回更新后的点赞数。

4. `addComment(Comment comment)`方法：

这个方法用于向数据库添加评论。参数是一个`Comment`对象，返回添加的行数。

- 首先获取数据库连接，并检查评论内容是否符合要求。

- 使用预处理语句将评论内容插入到数据库表中。

5. `deleteComment(int commentId, int userId)`方法：

这个方法用于删除评论。参数是评论ID和用户ID，返回表示删除是否成功的布尔值。

- 在`try-with-resources`语句块中获取数据库连接，并使用预处理语句删除具有指定commentId和userId的评论。

- 返回删除操作是否影响了数据库的行数。

6. `isCommentOwner(int commentId, int userId)`方法：

这个方法用于检查评论是否属于指定的用户。参数是评论ID和用户ID，返回布尔值。

- 在`try-with-resources`语句块中获取数据库连接，并使用预处理语句查询具有指定commentId的评论。

- 尝试获取结果集，并检查该结果集中的user\_id是否与指定的userId匹配。

总体来说，这段代码封装了一些与评论相关的数据库操作，包括点赞评论、添加评论、删除评论和检查评论所有者等功能。使用预处理语句和数据库连接来执行这些操作，确保了代码的安全性

这段代码定义了一个`Comment`类，用于表示评论对象。以下是对代码的分析和讲解：

1. 字段：

- `commentId`：评论ID

- `userId`：用户ID

- `animeId`：动画ID

- `content`：评论内容

2. 构造方法：

- 默认构造函数：没有参数的构造函数，未做任何操作。

- 带参构造函数：接受评论ID、用户ID和评论内容作为参数，并将其设置到对应的字段中。

3. Getter和Setter方法：

- `getCommentId()`和`setCommentId(int commentId)`：获取和设置评论ID。

- `getUserId()`和`setUserId(int userId)`：获取和设置用户ID。

- `getContent()`和`setContent(String content)`：获取和设置评论内容。

- `getAnimeId()`和`setAnimeId(int animeId)`：获取和设置动画ID。

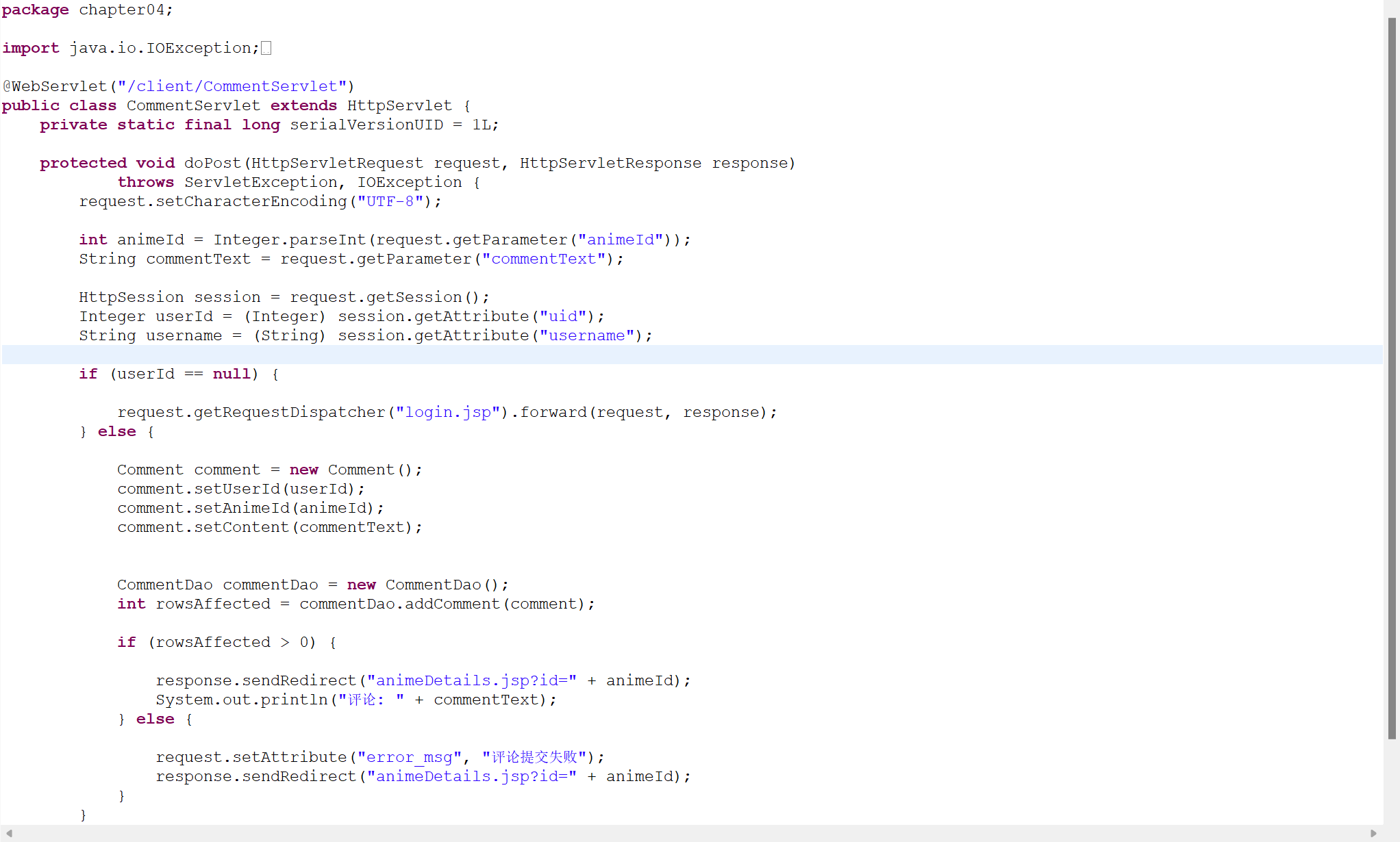
4. 静态方法：

- `isCommentValid(String content)`：根据特定的规则验证评论内容的有效性。

- 在这个例子中，验证的规则是评论内容的长度必须大于等于4个字符。

- 如果满足规则，返回true；否则，返回false。

总体来说，这个`Comment`类主要用于表示评论对象，并提供了相应的字段、构造方法和访问方法。此外，还提供了一个静态方法用于验证评论内容的有效性。

这段代码是一个`CommentServlet`类，是一个扩展了`HttpServlet`的Servlet类。它用于处理来自客户端的评论请求。

以下是对代码的分析和讲解：

1. `doPost()`方法：处理POST请求，获取评论相关的参数，并执行相应的操作。

- 首先通过`request.setCharacterEncoding("UTF-8")`设置请求字符编码为UTF-8，确保中文字符正确处理。

- 通过`request.getParameter("animeId")`获取动画ID。

- 通过`request.getParameter("commentText")`获取评论文本内容。

- 通过`request.getSession()`获取当前会话的`HttpSession`对象。

- 通过`session.getAttribute("uid")`和`session.getAttribute("username")`获取当前用户的ID和用户名。

- 如果`userId`为空，即用户未登录，跳转到登录页面`login.jsp`。

- 否则，创建一个`Comment`对象，设置评论相关的字段。

- 创建一个`CommentDao`对象，用于执行与评论相关的数据库操作。

- 调用`commentDao.addComment(comment)`执行添加评论的操作。

- 如果成功添加评论（`rowsAffected > 0`），重定向到`animeDetails.jsp`页面，并在URL中包含动画ID。

- 否则，将错误消息设置为请求属性`"error\_msg"`，然后重定向到`animeDetails.jsp`页面，并在URL中包含动画ID。

总体来说，这个Servlet类用于接收来自客户端的评论请求，检查用户是否已登录，然后创建一个`Comment`对象并将其添加到数据库。根据操作结果，将用户重定向到相应的页面，并在必要时提示错误消息。