

本节主要解决在详情页根据文章ID查找文章内容的问题。

1.根据ID查询文章数据

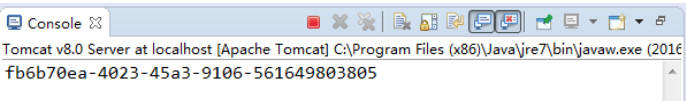
在上一节中，我们已经成功在detail.jsp页面获取到了来自index.jsp的文章ID。

那么，最容易想到的办法，就是直接在detail.jsp页面通过Java代码，直接查询出对应的文章数据，然后放到页面作用域就OK了。

代码：

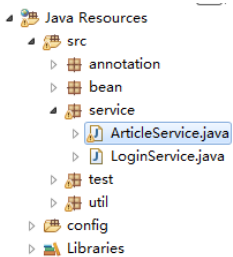
```
<%
    String id = request.getParameter("id");
    System.out.println(id);
%>
```

随便在首页点击一篇文章，然后看一下效果：



果然，可以拿到首页传过来的ID了。

对应的服务程序



```
/**
 * 通过文章id获取文章内容
 * @param id
 * @return
 */
public List<Map<String,Object>> getContentByArticleId(String id){
    String sql = "select content from t_article where id = ?";
    return DataBaseUtils.queryForList(sql,id);
}
```

我们直接调用这个方法。

额，不对，等等，这个方法好像有点问题，我当初怎么会这么写着来着？

因为文章的ID是唯一的，一个ID肯定对应一篇特定的文章。另外，我们不仅仅要查询文章内容，还需要文章的发布时间和分类等信息。

改一下：

```
/**
 * 通过文章id获取文章内容
 * @param id
 * @return
 */
public Map<String,Object> getContentByArticleId(String id){
    String sql = "select * from t_article a inner join t_category b on a.category_id = b.category_id where a.id = ?";
    return DataBaseUtils.queryForList(sql,id).get(0);
}
```

好吧，小细节不用在意。

继续，有了后台方法，就相当于稳定了大后方。接着我们就可以直接在jsp页面调用这个后台方法了。

```
<%
String id = request.getParameter("id");
Map<String,Object> map = articleService.getContentByArticleId(id);
pageContext.setAttribute("article", map);
%>
```

内容区的数据也要全部换成动态的：

```
<!-- 内容区 -->
<div class='article'>
    <div class='title'>${article.name}</div>
    <div class='category'><span class='light-font'>分类:</span><span class='info'>${article.category_name}</span></div>
    <div class='publicDate'><span class='light-font'>发布时间:</span><span class='info'>${article.create_time}</span></div>
    <hr/>
    <div class='content'>
        ${article.content}
    </div>
</div>
```

然后，重启tomcat，在首页随便点开一篇文章，就能看到效果了。



JavaScript闭包详解

分类：编程代码类

发布时间：2016-10-19 10:43:10.0



作者：张三

本文面向初学者，大神轻喷。

闭包是什么？

初学javascript的人，都会接触到一个东西叫做闭包，听起来感觉很高大上的。网上也有各种五花八门的解释，其实我个人感觉，没必要用太理论化的观念来看待闭包。

事实上，你每天都在用闭包，只是你不知道罢了。

比如：

```
var cheese = '奶酪';

var test = function(){
    alert(cheese);
}
```

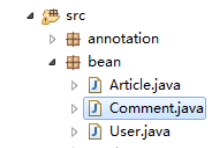
这些数据已经不是静态的了，而是从数据库里面查询出来的。

2.评论功能后台业务实现

文章的信息查询已经没问题了，接下来，就是评论功能了，我们先把评论的后台方法都写好。

2.1 保存评论

有了之前章节的基础，这一步应该还是比较简单了。首先，建立JavaBean。



```
package bean;

import annotation.Column;
import annotation.Table;

@Table(tableName = "t_comment")
public class Comment {

    @Column(type = "varchar(100)", field = "id", primaryKey = true, defaultNull = false)
    private String id; //主键，采用UUID

    @Column(type = "VARCHAR(100)", field = "user_id")
    private String userId; //评论者的ID
```

```

@Column(type = "VARCHAR(600)", field = "content")
private String content; //评论内容

@Column(type = "VARCHAR(100)", field = "article_id")
private String articleId; //文章ID

@Column(type = "datetime", field = "create_time")
private String createTime; //创建时间

@Column(type = "timestamp", field = "update_time")
private String updateTime; //最后更新时间

@Column(type = "int(1)", field = "is_delete")
private Integer isDelete; // 删除状态 0未删除 1删除

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getUserId() {
    return userId;
}

public void setUserId(String userId) {
    this.userId = userId;
}

public String getContent() {
    return content;
}

public void setContent(String content) {
    this.content = content;
}

public String getCreateTime() {
    return createTime;
}

public void setCreateTime(String createTime) {
    this.createTime = createTime;
}

public String getUpdateTime() {
    return updateTime;
}

public void setUpdateTime(String updateTime) {
    this.updateTime = updateTime;
}

public Integer getIsDelete() {
    return isDelete;
}

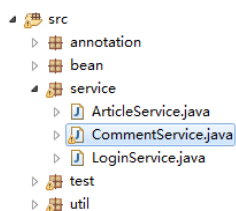
public void setIsDelete(Integer isDelete) {
    this.isDelete = isDelete;
}

public String getArticleId() {
    return articleId;
}

public void setArticleId(String articleId) {
    this.articleId = articleId;
}
}

```

然后是评论服务类:



先是保存评论的方法:

```

/**
 * 保存评论
 */
public void saveComment(Comment comment) {
    String sql = "insert into t_comment(id,user_id,content,article_id,create_time,is_delete) values(?,?,?,?,?,?)";
    DataBaseUtils.update(sql, comment.getId(), comment.getUserId(),
        comment.getContent(), comment.getArticleId(), new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(new Date()), 0);
}

```

测试:

测试的话, 我们给《JavaScript闭包详解》这篇文章, 添加一条张三的评论吧。

```

CommentService cs = new CommentService();
Comment comment = new Comment();
comment.setId(UUID.randomUUID().toString());
comment.setContent("很不错的文章, 赞一个!");
comment.setArticleId("fb6b70ea-4023-45a3-9106-561649803805");
comment.setUserId("319600c3-550a-4f9f-80cf-deebe2376528");
cs.saveComment(comment);
System.out.println("保存成功!");
System.out.println(DataBaseUtils.queryForList("select a.content from t_comment a left JOIN t_user b "

```

```
+ "on a.user_id = b.id where a.article_id = 'fb6b70ea-4023-45a3-9106-561649803805'"));
```

先保存一条数据，然后再把评论信息查询出来。

效果：

```
保存成功!
[{"content=很不错的文章，赞一个! "}]
```

这样就OK了。

2.2 查询评论

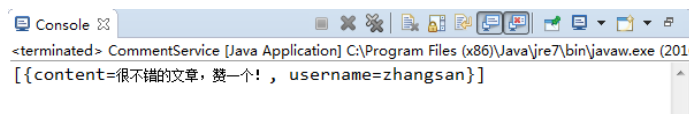
然后是查询评论的方法：

```
/**
 * 根据文章id查询它的所有评论
 * @param id
 * @return
 */
public List<Map<String,Object>> getCommentsByArticleId(String id){
    return DataBaseUtils.queryForList("select b.username ,a.content from t_comment a left JOIN t_user b " +
        "on a.user_id = b.id where a.article_id = ?", id);
}
```

测试：

```
CommentService cs = new CommentService();
System.out.println(cs.getCommentsByArticleId("fb6b70ea-4023-45a3-9106-561649803805"));
```

效果：



```
<terminated> CommentService [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (201
[{"content=很不错的文章，赞一个! , username=zhangsan}]
```

嗯，差不多可以了。

源码地址：<http://pan.baidu.com/s/1nhN754s#list/path=%2F>

我的个人博客：<http://s-335245.gotocdn.com8080/index.html>