【手把手】JavaWeb 入门级项目实战 -- 文章发布系统 （第十节）

本节主要讲解详情页的页面布局和css样式，以及从主页面到详情页面的跳转问题。

1. 详情页面的布局

页面布局的话，我还是习惯先把静态页面做出来，确保没问题了，然后再跟后台对接。

在实际的开发过程中，静态页面一般是美工或者前端工程师负责，后台工程师主要关心如何把Java层的数据贴到静态页面。（虽然我经历的几个公司都是自己一个人全包了。。。）

之前我已经把我个人绘制静态页面的过程，还有自己的思路写出来了，所以从本节开始，我不会再写得那么详细，而是根据页面效果来简明扼要地介绍一下。

我绘制页面的一般流程，就是先在脑子里形成一个大概的模样，就是搞清楚到底要画一个怎样的页面？甚至还可以弄个草图，然后根据草图来逐步写html和css代码。

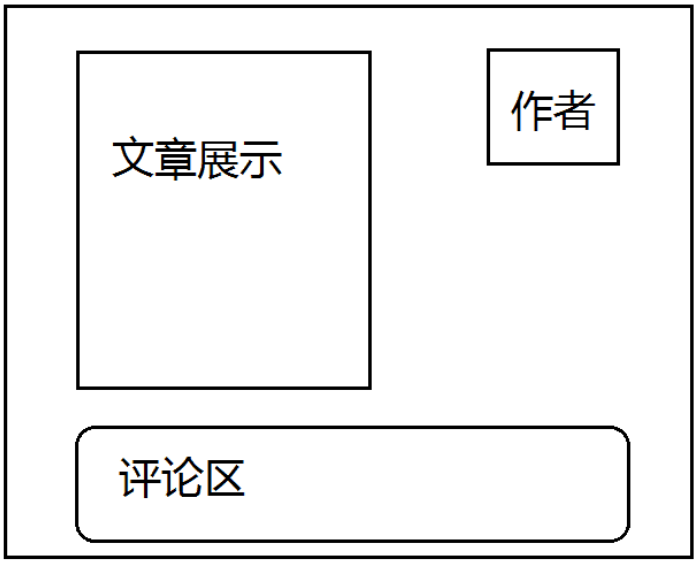
首先是搭骨架，骨架就是html代码。

搭好了骨架，才是添加css代码的时候。

我个人绘制页面，都是遵循这样一个流程。

好了，先看看效果：

页面布局简图：



第一部分，是文章的展示区域。

html骨架

```
<!-- 内容区 -->
<div class='article'>
```

```
    <div class='title'>文章标题</div>
    <div class='category'><span class='light-font'>分类：</span><span class='info'>编程代码类</span></div>
    <div class='publicDate'><span class='light-font'>发布时间：</span><span class='info'>2016-10-27</span></div>
    <hr/>
    <div class='content'>
        <p>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。

        <p>事实上，你每天都在用闭包，只是你不知道罢了。

        <p>比如：

        var cheese = '奶酪';

        var test = function(){
            alert(cheese);
        }
        OK，你已经写了一个闭包。

        <p>函数也是一个数据类型</p>
        <p>变量 cheese 是在全局作用域中的一个变量，当你创建了一个 test 函数，那么，test 和 cheese 就共享一个全局作用域。</p>

        <p>你要额外明白的一点是，在js中，函数和变量本质上是一个东西。函数也是一个数据类型。</p>

        <p>从上面的定义中也能看出来这一点。你要是不相信的话，我们来看一下咯。</p>

        <p>alert(cheese);</p>
        <p>alert(test);</p>

        <p>Paste_Image.png</p>

        <p>Paste_Image.png</p>
        <p>让我们再来看看 test 和 cheese各是什么类型：</p>

    </div>
</div>
```

文章内容都是一些**测试数据**。

展示一篇文章，包含这篇文章的标题，分类和发布时间，当然还有作者信息。

首先是文章标题，它就是一个div，没什么大不了的。不要把它想得太复杂了。

```
<div class='title'>文章标题</div>
```

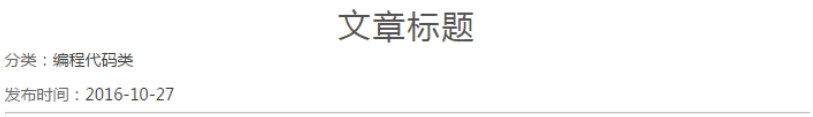
标题的css：

```
.article .title {
    text-align: center;
    font-size: 28px;
    color: #565353;
    letter-spacing: 2px;
    margin-top:20px;
}
```

文字居中：

```
text-align: center;
```

这句话能让这个div里面的文字居中显示，所以你才能看到这个居中效果：



标题的字体肯定不能太小了，所以我先给它一个28px。

```
font-size: 28px;
```

我觉得字体太黑了不好看，就把颜色稍微调淡了一些。

```
color: #565353;
```

接着是字间距，我们不希望每个字都紧凑得挤在一起，所以让字与字之间稍微空开一点。

```
letter-spacing: 2px;
```

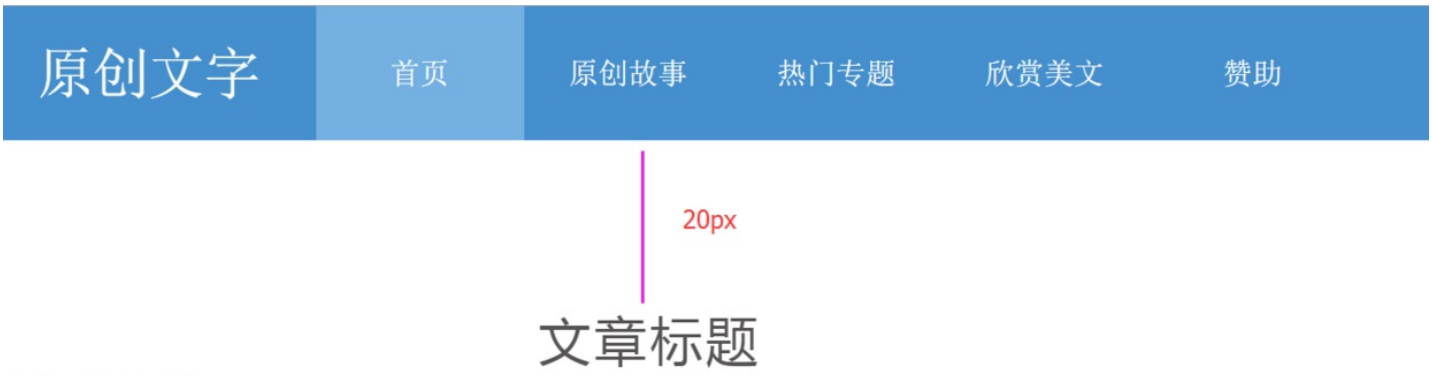
这表示空开2个px。

letter-spacing真的可以做到，有图为证。

向上的间距也要调大一点，不然紧紧贴着标题栏肯定不好看。

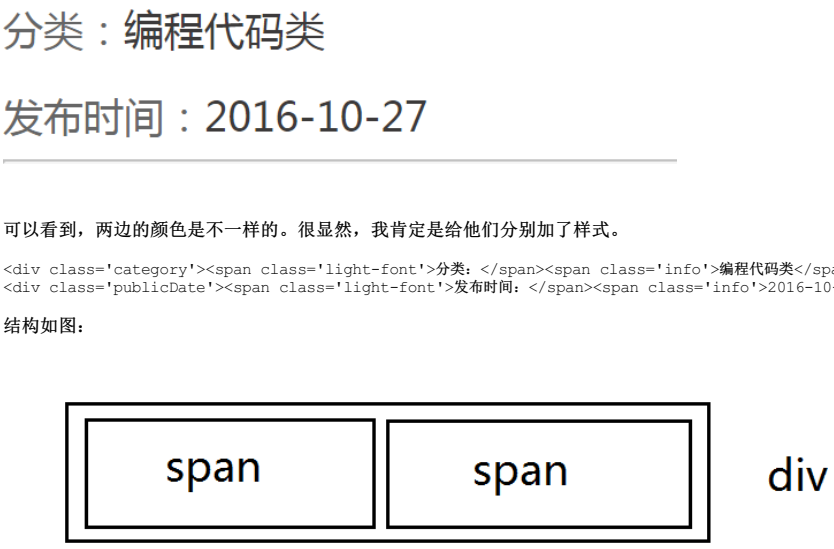
```
margin-top:20px;
```

就20个px吧。



这样就差不多了。

接下来是这一块。



可以看到，两边的颜色是不一样的。很显然，我肯定是给他们分别加了样式。

```
<div class='category'><span class='light-font'>分类：</span><span class='info'>编程代码类</span></div>
<div class='publicDate'><span class='light-font'>发布时间：</span><span class='info'>2016-10-27</span></div>
```

结构如图：

为了分别控制左右两边的字体样式，我在外层div中嵌套了两个span，给他们分别加上不同的样式。

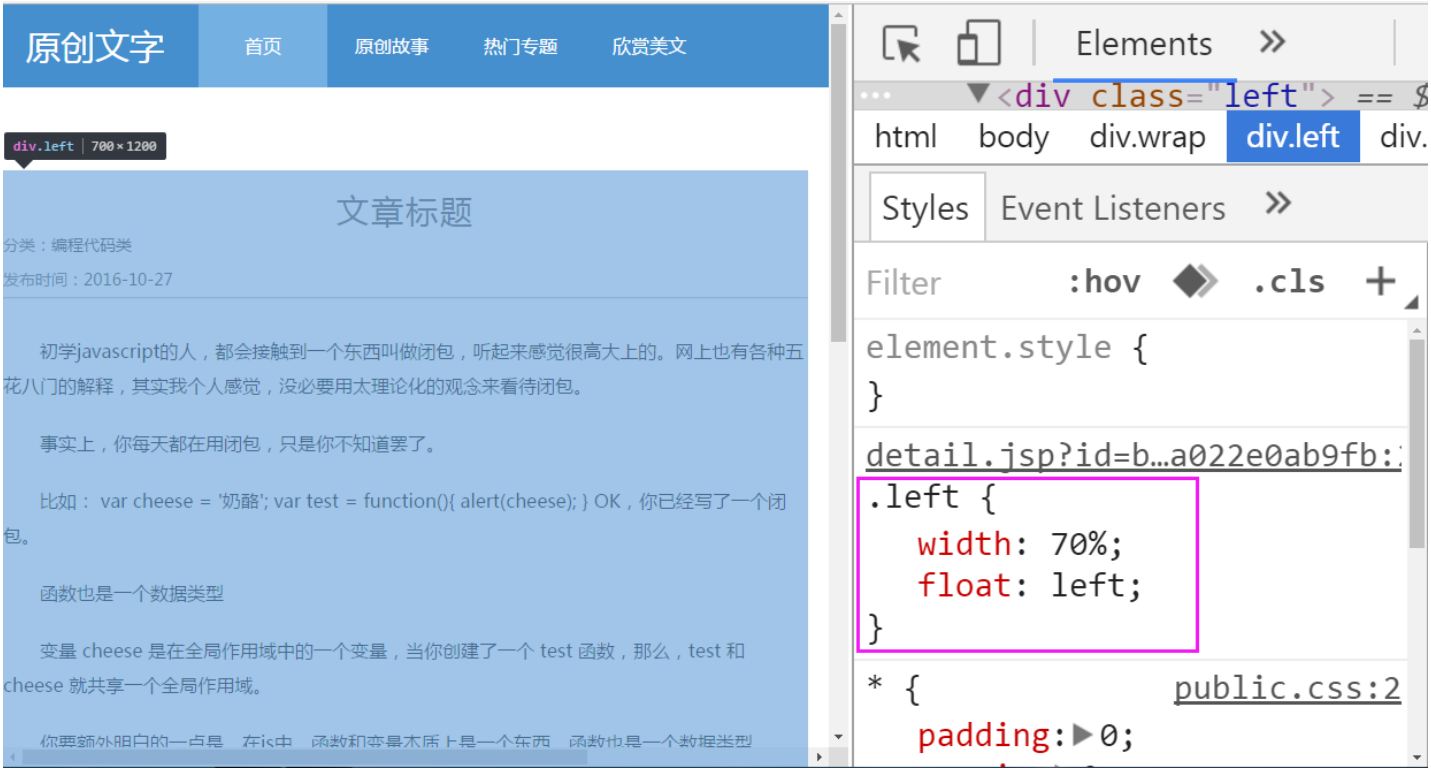
对应的css样式

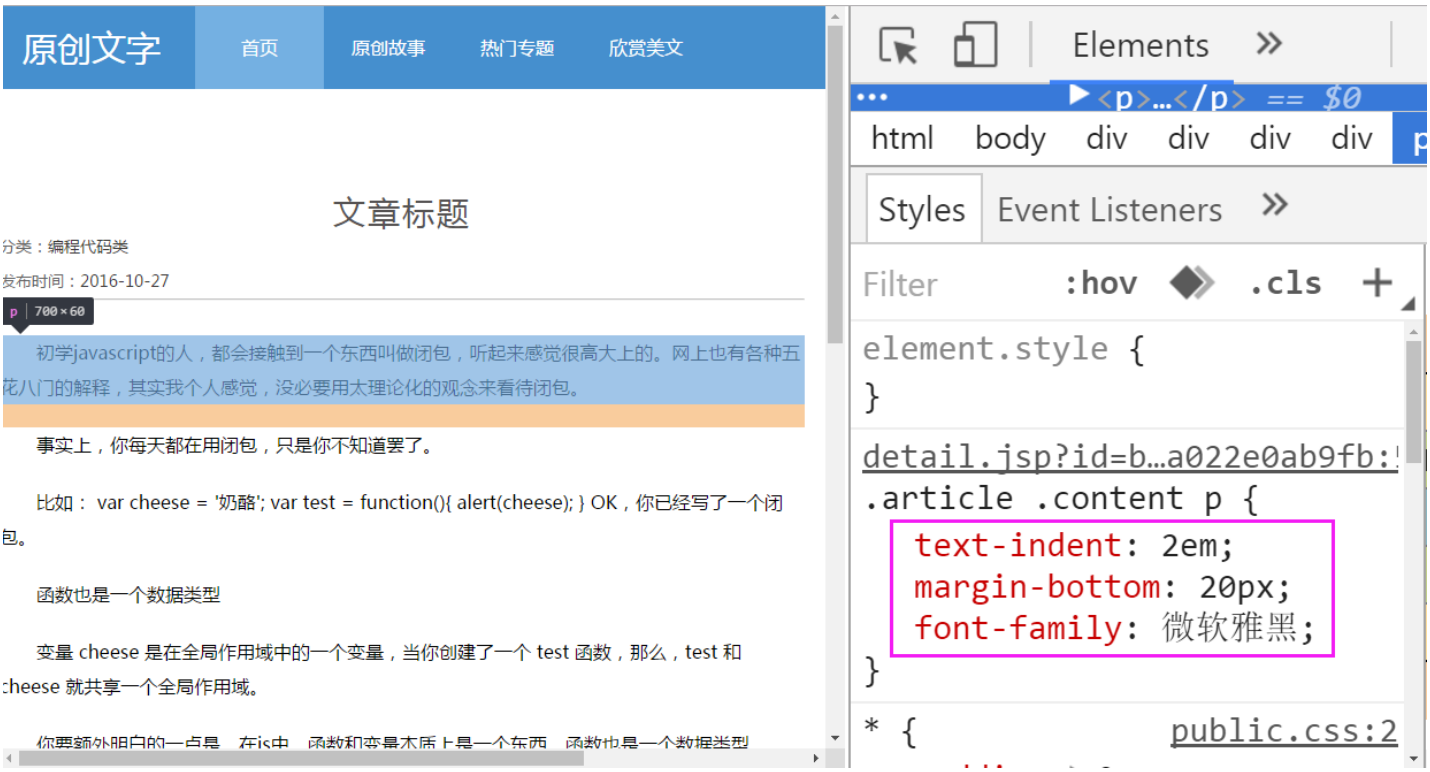
```
.article .light-font{
    font-size:14px;
    color:#666;
}

.article .info{
    font-size:14px;
    color:#3c3a3a;
}
```

文章区域，我主要给里面的 P 标签加了样式：

```
.article .content p{
    text-indent:2em;
    margin-bottom:20px;
    font-family: 微软雅黑;
}
```

接下来演示一下层级关系：





文章中的每一个段落，对应一个P标签，我给这些P标签加上特定的样式，包括首行缩进，底外边距，还有字体。

先这么简单弄一下吧，有个样子就行了。

至于作者信息的展示区，也是比较简单的，我就放了一个头像和作者名称。


作者：张三

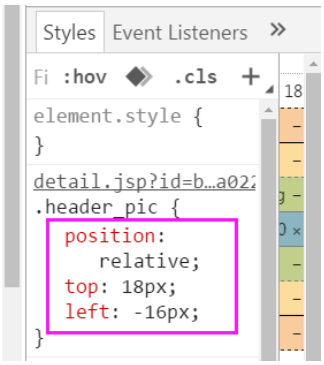
html

```
<div class='right mt32'>
    <div class='author'>
    <img src='${basePath}/static/img/1.jpg' class='header_pic' width='90' height='90'></img>
    作者：张三
    </div>
</div>
```

可见，头像和作者名称都是放在一个div里面的，这个div有一个叫做 right 的css属性。

头像部分我用了一个相对定位

作者：张三



最后是评论区：

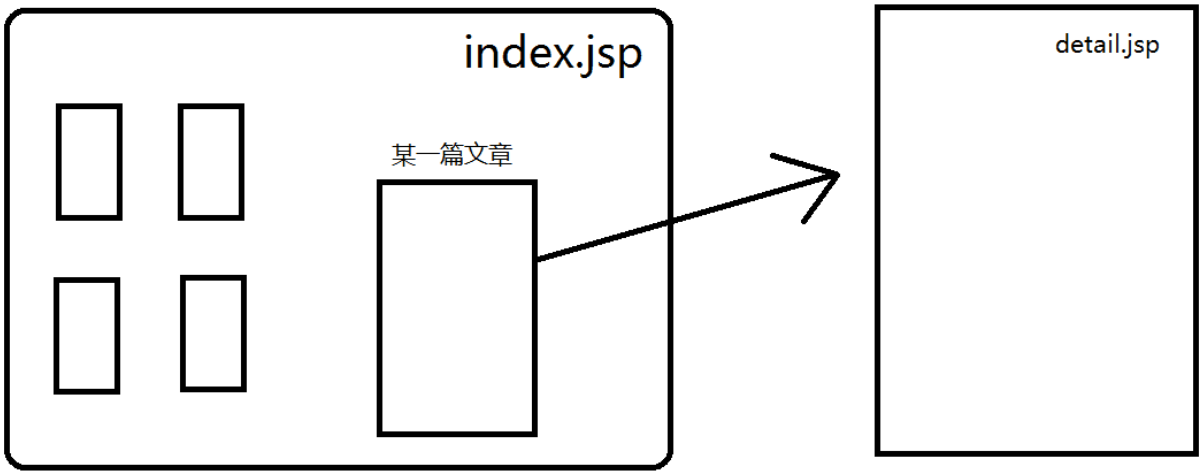
剩下的具体评论是一个个p标签，就不一一细说了。

评论区html代码：

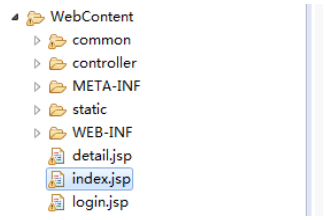
```
<!-- 评论区 -->
<div class='commentBox'>
    <textarea class="comment_input" id="commenttxt" placeholder="请输入评论信息(600)..." maxlength="600"></textarea>
        <input type="button" value="保存评论" class="button">
</div>
<div class='clearfix'></div>
<br/><hr/>
<div class='mb64' class="comment_list">
    <div class="comment_infor clearfix">
        <div style='border-bottom:1px solid #ccc' class="comment_word">
            <p style='border-bottom:20px solid #fff'>${comment.username}dwedewffrg 说：</p>
            <p class='mb32'>内容不错，感谢分享！</p>
        </div>
    </div>
</div>
```

2. 从主页面到详情页面的跳转问题

我们的首页会展示出不同分类的文章列表，每一篇文章都有一个封面，我们通过点击封面进入文章的详情页。



这个时候，我们先来看看，当初是怎么把数据库里面的文章展示在首页的？



原来，我们在index.jsp中，写过如下代码：

```
<% ArticleService articleService = new ArticleService(); %>
```

然后调用了 **articleService** 的 **getArticlesByCategoryId** 方法。

```
<%
    //查询出编程代码类的相关文章
    List<Map<String,Object>>  articles2 = articleService.getArticlesByCategoryId(2, 0, 6);
    pageContext.setAttribute("articles2", articles2);
%>
```

默认查出来前六条数据。

如果没看明白的话，可以去回顾之前的章节，这里就不再赘述了。

好的，继续。

articles2 是一个list，拿到这个list之后，我们立即把它放在了pageContext里面，这样的话，我们在当前页面就能够访问到这个list了。

像这样：

```
<div class='category'>
    <div class='title'>编程代码类</div>
    <ul class='items'>
        <c:forEach items="${articles2}" var="item">
            <li class='item' onclick="detail('${item.id}');">
                <div class='item-banner'>
                    <div class='item-header'>${item.header}</div>
                    <div class='item-name' title = "${item.name}">${item.name}</div>
                    <div class='item-author'>@${item.author} 著</div>
                </div>
                <div class='item-description'>${item.description}</div>
            </li>
        </c:forEach>
        <div style='clear:both'></div>
    </ul>
</div>
```

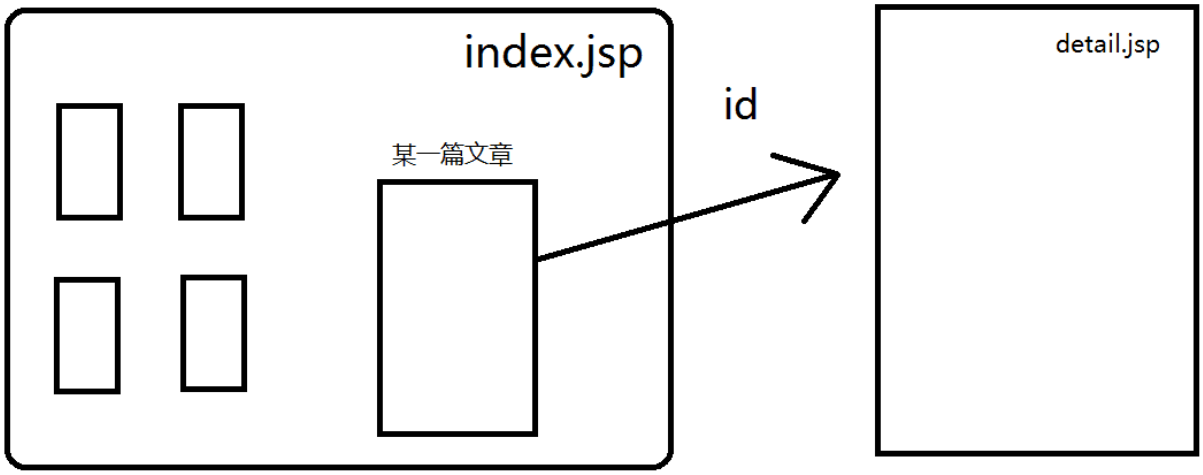
我们要通过这个封面打开详情页，是不是需要一个点击事件呢？

因为我这边使用了onclick属性，所以我单独写了一个 detail 方法：

```
//打开详情页
function detail(id){
    var a = document.createElement("a");
    a.href = "detail.jsp?id=" + id;
    console.log(a);
    a.target = '_new'; //指定在新窗口打开
    a.click();//触发打开事件
}
```

在detail方法里，我直接创造了一个a标签，目标地址指向了detail.jsp，并且使用get方式传递了一个参数，也就是文章的id。最后，手动触发了点击事件。

因为需要在detail.jsp中，从后台查出文章的具体内容，我们必然通过id去查。所以，我们需要给详情页传递一个id。



我们这一章先不管怎么去后台查，先确保能把文章id传递到详情页再说。

当我们点击文章列表中的某一条数据，进入详情页的时候，会发现url地址栏就带了id。

比如：
http://localhost:8080/Article/detail.jsp?id=ddc0136f-7bf5-41ed-ba6f-462d370afbe4

点击不同的文章，后面跟的id是不同的。这样就说明id的传递已经没问题了。

好的，这一章先到这里，下一节开始写详情页的业务代码。

额，家里渣网速，怎么也打不开 FTP，过几天我再上传本章的源码吧。