“一起恋爱吧”功能文档

郝伟博

张春龙

杨志雨

目录

loveyou 是一个时尚交友平台，我们用 django 框架加上 sqlite3 数据库进行开发的。

bootstrap 优化前台网页技术

网站前台美化利用了 bootstarap，在头部文件导入三个文件：

<script src="/js/jquery-1.9.1.js"></script> --导入 juquery

<link href="/css/bootstrap.min.css" rel="stylesheet" media="screen"> --导入 bootstrap 的基本 css 样式

<script src="/js/bootstrap.min.js"></script> --导入 bootstrap 的 js 脚本

1.下载编译好的文件

2.bootstrap 文档结构：

```
bootstrap/
├── css/
│   ├── bootstrap.css                编译好的 js 和 css 文件
│   ├── bootstrap.min.css            编译好压缩后的 js 和 css 文件
├── js/
│   ├── bootstrap.js
│   ├── bootstrap.min.js
└── img/
    ├── glyphicons-halflings.png
    └── glyphicons-halflings-white.png    图片使用 imageoptim 进行压缩
```

所有的 JavaScript 插件都依赖 jQuery 库

3..bootstarp 文件的导入

1.<link href='bootstarp/css/bootstrap.min.css' rel='stylesheel' media='screen'/>

2.js 文件的导入

```
<script  src='jqurey.min.js'></script>
<script src='bootstrap/js/bootstrap.min.js'></script>
```

3.Bootstrap 使用的某些 HTML 元素和 CSS 属性需要文档类型为 HTML5 doctype。因此这一文档类型必须出现在项目的每个页面的开始部分。

```
<!DOCTYPE html>
<html lang="en">
  ...
</html>
```

4.Bootstrap 默认的栅格系统为 12 列 ，形成一个 940px 宽的容器，默认没有启用 响应式布局特性 。如果加入响应式布局 CSS 文件，栅格系统会自动根据可视窗口的宽度从 724px 到 1170px 进行动态调整。在可视窗口低于 767px 宽的情况下，列将不再固定并且会在垂直方向堆叠。

1.对于简单的两列式布局

1.创建一个（.row）的容器

2.在容器中加入适量的（.span*）即可

由于默认 12 列的栅格（.span*）列最多不能超过 12

```
<div class="row">
    <div class="span4">...</div>
    <div class="span8">...</div>
</div>
```

上面的代码展示了 .span4 和 .span8 两列，两列的和总共是 12 个栅格

2.对于简单的布局：两列之间还应该有间距则这个间距用（.offset）来表示

```
<div class="row">
    <div class="span4">...</div>
    <div class="span3 offset2">...</div>
</div>
```

5.bootstrap 的流式栅格系统对每一列的宽度使用百分比而不是像素数量。它和固定栅格系统一样拥有响应式布局的能力，这就保证它能对不同的分辨率和设备做出适当的调整。

1.将 .row 替换为 .row-fluid 就能让任何一行“流动”起来。应用于每一列的类不用改变, 这样能方便的在流式与固定栅格之间切换。

```
<div class="row-fluid">
    <div class="span4">...</div>
    <div class="span8">...</div>
</div>
```

2.流式栅格的间距：

```
<div class="row-fluid">
    <div class="span4">...</div>
    <div class="span4 offset2">...</div>
</div>
```

6.布局

1.提供了一个通用的固定宽度(也可以变为响应式)的布局方式，仅仅用 即可。（类似居中固定大小）

```
<body>
<div class="container">
  ...
</div>
</body>
```

流式布局

2.利用 代码可以创建一个流式、两列的页面 — 非常适合于应用和文档类页面。 （类似居中以百分比固定大小）

```
<div class="container-fluid">
<div class="row-fluid">
<div class="span2">
<!--Sidebar content-->
</div>
<div class="span10">
<!--Body content-->
</div>
</div>
</div>
```

7.启用响应式特性

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link href="assets/css/bootstrap-responsive.css" rel="stylesheet">
```

8.bootstrap 中 css 操作：

1.初始值：Bootstrap 定义的全局 font-size 是 14px，line-height 是 20px。这些样式应用到了 <body> 和所有的段落上。另外，对 <p> (段落)还定义了 1/2 行高(默认为 10px)的底部外边距(margin)属性。

<p>的 class 的操作：

Make a paragraph stand out by adding (.lead)
Fusce dapibus, tellus ac cursus commodo, tortor mauris nibh.
正常
Etiam porta sem malesuada magna mollis euismod. 警告
Donec ullamcorper nulla non metus auctor fringilla. 错误
Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis. 信息
Duis mollis, est non commodo luctus, nisi erat porttitor ligula. 成功

2.缩略语:

<abbr title="attribute">attr</abbr> attr 是 title 里面的那个的缩略词

<addr title =' 添加的小字符'class = 'initialism'>attr</addr> 为 <abbr> 标签添加 .initialism 类使其使用更小一些的字号

3.地址：

```
<address>
<strong>Twitter, Inc.</strong><br>
795 Folsom Ave, Suite 600<br>
San Francisco, CA 94107<br>
<abbr title="Phone">P:</abbr> (123) 456-7890
</address>

<address>
<strong>Full Name</strong><br>
<a href="mailto:#">first.last@example.com</a>
</address>
```

4.blockquote 的用法

```
<blockquote class="pull-right">
...
</blockquote>
```

文件到右边 相反的是 pull-rigth

5.列表（ul 和 ol）

```
<ul class="unstyled">
<li>...</li>
</ul>
```

去掉列表的样式

6.表格的样式

```
<table class='table'>
<tr>
<td>aaaaaaaaaaa</td>
<td>bbbbbbbbbbbb</td>
</tr>
<tr>
<td>aaaaaaaaaaa</td>
<td>bbbbbbbbbbbb</td>
</tr>
</table>
```

还有更多样式 table table-striped 边框（border）和圆角（rounded corner）

7.为 <table> 标签增加基本样式--很少的内补(padding)并只增加水平分隔线--只要为其增加 .table 类即可

```
<table class="table">
…
```

```
</table>
```

8.通过 :nth-child 选择器为 <tbody> 中包含的每一行增加条状斑马纹样式 (IE7-IE8 不支持)。

```
<table class="table table-striped">
  ...
</table>
```

9.为表格增加边框(border)和圆角 table table-bordered

```
<table class="table table-bordered">
  ...
</table>
```

10.为 <tbody> 中的每一行赋予鼠标悬停样式 table table-hover

网页包括网站首页，我的约会，搜索约会，约会广场，登录注册

juqery 动态加载图片效果

```
<script src="js/jquery-1.8.2.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/jquery.bxSlider.min.js"></script>
    <script type="text/javascript">
        $(document).ready(function () {
            $('#slider').bxSlider({
                auto: true,
                pager:true
            });
            $('#slider2').bxSlider({
                mode: 'fade',
                captions: true,
                auto: true,
                controls: false
            });
        });
    </script>
```

只要在 div 中插入图片就滚显示滚动效果：

```
<div class="pd20">
                <ul id="slider">
                    <li><a href="#"><img src="img/s1.jpg" alt="" /></a></li>
                    <li><a href="#"><img src="img/s2.jpg" alt="" /></a></li>
                    <li><a href="#"><img src="img/s3.jpg" alt="" /></a></li>
                    <li><a href="#"><img src="img/s3.jpg" alt="" /></a></li>
                </ul>
</div>
```

文件路径问题：

如果在 svn 协同开发的时候，当输入文件路径不一致的情况下，只要 settings.py 中对 media_root 进行修改：

```
MEDIA_ROOT = os.path.join(os.path.dirname(__file__), './mymedia').replace('\\', '/')
```

如何加载静态文件问题

```
url(r'js/(?P<path>.*)$',
'django.views.static.serve',{'document_root':os.path.join(os.path.dirname(__file__),"./templates/js")},name='js'),
url(r'css/(?P<path>[\w\.\-]+\.css)$',
'django.views.static.serve',{'document_root':os.path.join(os.path.dirname(__file__),"./templates/css")},
```

```
name='css'),
url(r'img/(?P<path>[\w\.\-]+\.*)$',
'django.views.static.serve',{'document_root':os.path.join(os.path.dirname(__file__),"./templates/img")}
,name='img'),

    url(r'media/(?P<path>[\w\.\-]+\.*)$',
'django.views.static.serve',{'document_root':os.path.join(os.path.dirname(__file__),"./mymedia")},na
me='media'),
```

这样的好处就是可以在 template 中引入 js，css，img

models

首先，在公共起步的时候，在建数据库模型的时候反反复复的出现各种问题，最主要的是数据库模型建的不合理。

在约会这块，出现的问题最多。

```
#约会类的
class YueHui(models.Model):
    xuser = models.ForeignKey(User)
    sort = models.ManyToManyField(u'Sort')
    content = models.TextField(max_length = 200,verbose_name = '内容',null = True)
    r_time = models.DateTimeField(auto_now=True)#时间
    class Meta:
        ordering = ['-r_time']
    def __unicode__(self):
        return self.content
```

主要还是在处理分类和约会表的关系上。

在建好数据库模型的时候，按部就班的开始下面的工作了。

配置相关的文件倒是没有什么大的问题。在写视图函数的时候，由于思路不清晰和对数据库数据的技术不太熟练，导致出现了一些比较尴尬的场面。

例如，在获取的时候发现没有这个方法，或者，有这个方法系统却提示没有。弄得很郁闷。后来才测试后才发现，原来是没有同步数据库，导致在修改了 model 的时候，字段没有添加进去。

首先，说一下写功能函数的问题。这是一个比较大的问题。在写视图函数的时候，思路不清晰会对写函数有很大的影响。这会使写的功能函数达不到预想的效果，或者根本就不对。

```
def index_myperson(request):
    def index_myperson(request):
    user = request.user
    yuehuis = user.yuehui_set.all()
    context = {'user':user}
    return render_to_response('index_myperson.html',context)
def index_mydate(request):
    user = request.user
    yuehui = user.yuehui_set.all()
    context = {'user':user,'yuehui':yuehui}
    return render_to_response('index_mydate.html',context)

def index_myfollow(request):
    '''关注'''
    user = request.user
    context = {'user':user}
    return render_to_response('index_myfollow.html',context)
def index_mycollect(request):
    '''收藏'''
```

```
    user = request.user
    collect = user.get_profile().collect.all()
    collect_num = len(collect)
    context = {'user':user,'collect_num':collect_num,'collect':collect}
    print 'aaaaaaaaa'
    return render_to_response('index_mycollect.html',context)

    yuehuis = user.yuehui_set.all()
    context = {'user':user}
    return render_to_response('index_myperson.html',context)
def index_mydate(request):
    user = request.user
    yuehui = user.yuehui_set.all()
    context = {'user':user,'yuehui':yuehui}
    return render_to_response('index_mydate.html',context)
```

后台公共方法

```
def index_myfollow(request):
    '''关注'''
    user = request.user
    context = {'user':user}
    return render_to_response('index_myfollow.html',context)
def index_mycollect(request):
    '''收藏'''
    user = request.user
    collect = user.get_profile().collect.all()
    collect_num = len(collect)
    context = {'user':user,'collect_num':collect_num,'collect':collect}
    print 'aaaaaaaaa'
    return render_to_response('index_mycollect.html',context)
```

上面是我写的功能函数，尤其是在获取当前用户的数据上，很是费了很的大时间才搞明白。
一开始的时候，用的是 id 传值，结果效果不是很好，在 html 中路径写的有问题，导致不能自动获取 id。
后来用的是 def index_myperson(request):
    user = request.user
这个方法，就不用 id 传值就可获取到当前用户的数据。

后台搜索查询功能：

```
class SForm(forms.ModelForm):
    '''搜索表单'''
    class Meta:
        model = Sort
        fields = ('city','type','time','sexs')
def search(request):
    user1 = request.user
    user = User.objects.all()
    sort = Sort.objects.all()
    if request.method == 'POST':
        sform = SForm(request.POST)
        if sform.is_valid():
            city = sform.cleaned_data['city']
```

```
                type = sform.cleaned_data['type']
                time = sform.cleaned_data['time']
                sexs = sform.cleaned_data['sexs']
                sorts = sort.filter(city=city).filter(type=type).filter(time=time).filter(sexs=sexs)
                return
render_to_response('search.html',{'sform':sform,'sorts':sorts,'request':request,'user':user,'user1':user1})

#                 user2 = user1.all().filter(type=type)
#                 user3 = user2.all().filter(time=time)
#                 user4 = user3.all().filter(sexs=sexs)
#                                                                                                 user5        =
user4.user.exclude(username=request.user.username).exclude(username='root')
        else:
            sform = SForm()
```

约会广场部分功能：

约会广场分为两个部分：显示约会和创建约会

```
class LoginForm(forms.Form):
    username = forms.CharField(label="姓名")
    password = forms.CharField(label="密码",widget=forms.PasswordInput)
class CreateForm(forms.ModelForm):
    class Meta:
        model = Sort
        fields = ('city','type','time','sexs')
def gc(request):                #创建约会
    if request.method == 'POST':
        createform = CreateForm(request.POST)
        if createform.is_valid():
            user = createform.cleaned_data['user']
            city = createform.cleaned_data['city']
            type = createform.cleaned_data['type']
            time = createform.cleaned_data['time']
            sexs = createform.cleaned_data['sexs']
            sort = Sort.objects.create(user='user',city='city',type='type',time='time',sexs='sexs')
            return HttpResponse('aa')
    else:
        createform = CreateForm()
    return render_to_response('gc.html',{'createform':createform,'request':request})
def disp(request):        #显示当前的约会
    """广场"""
    yuehuis = YueHui.objects.all()
    users = User.objects.exclude(username=request.user.username).exclude(username='root')
    show_users = []
    for var_i in xrange(10):
        if users:
            user = random.choice(users)
            if user not in show_users:
                if request.user.is_authenticated():
                    if user not in request.user.get_profile().follow.all():
                        show_users.append(user)
                else:
                    show_users.append(user)
    yuehuis = YueHui.objects.all().order_by('-id')
```

```
loginform = LoginForm()
content = {
        'yuehuis':yuehuis,
        'request':request,
        'loginform':loginform,
        'show_users':show_users,
  }
return render_to_response('gc.html',content)
```

值得一提的是在前台实现的时候利用了一个 bootstrap 特效，点击创建约会就可以慢慢弹出一个窗口：

```
<div class="content">
    <a href="#myModal" role="button" data-toggle="modal" class="btn btn-large" id="cj">一起约会吧</a></p>
        <div id="myModal" class="modal hide fade" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>
                <h3 id="myModalLabel">亲  来创建个约会吧</h3>
            </div>
            <div class="modal-body">
                <form method="post" action="/home/myperson/">
                {{createform.as_p}}
                <input class="btn" type="submit" value="ok"></input>
                </form>          </div>
        </div>
</div>
```

结束语

这个项目做的时候时间很短，也出现了许多 bug，也有一些技术上的漏洞，后期我们会不断的优化和完善