第七节课

1. 配置好my_base.html页面
2. 在my_info.html里面引用my_base页面
3. 这个关于页面的继承.
4. 在my_info页面里面把所有回台的数据都传到前端
5. my_password页面也同理 还有其他的页面也填好 修改了img里面的数据, 其他的页面也跳调整好
6. 注意: 一些特殊的字段显示 img 和 选择类型的

```
# 显示图片的字段
{{ MEDIA_URL }}{{request.user.image}}

# 显示选择的类型的字段,使用一个if判断对其字段进行判断
{% if request.user.gende == 'male' %} checked="checked" {% endif %}
```

```
# 如果需要显示上传的图片需要设计url和setting
   #--> urls.py
   from django.views.static import serve

   from tanzhou.settings import MEDIA_ROOT

   #配置上传文件的访问处理函数
       url(r'^media/(?P<path>.*)$',  serve, {"document_root":MEDIA_ROOT}),

 #--> setting.py

TEMPLATES = [
    {
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.media'  # 这个是用来配置media的路径
            ]
    }
]

   MEDIA_URL = '/media/'
   MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

6. 在页面设置成form表单,

```
<form class="clearfix" id="jsAvatarForm" enctype="multipart/form-data" autocomplete="off"
method="post" action="{% url 'i:image_upload' %}" target='frameFile'>
                                        <label class="changearea" for="avatarUp">
                                            <span id="avatardiv" class="pic">
                                                <img width="100" height="100" class="js-
img-show" id="avatarShow" src="{{ MEDIA_URL }}{{ request.user.image }}"/>
                                            </span>
                                            <span class="fl upload-inp-box" style="margin-
left:70px;">

                                                <span class="button btn-green btn-w100"
```

```
id="jsAvatarBtn">修改头像</span>
                                                    <input type="file" name="image"
id="avatarUp" class="js-img-up"/>
                                                    <input class="_2qCjw" id="jsLoginBtn"
type="submit" value="确定" />

                                                </span>
                                            </label>
                                            {% csrf_token %}
                                        </form>
```

7. url.py里面地址

```
 # 用户头像上传
    url(r'^image/upload/$', UploadImageView.as_view(), name="image_upload"),
```

8. 在view.py里面设置提交照片的逻辑

```
# 用户修改头像
class UploadImageView(LoginRequiredMixin, View):
        # 把前段传入的数据保存
    def post(self, request):
        image_form = UploadImageForm(request.POST, request.FILES)
        if image_form.is_valid():
            image = image_form.cleaned_data['image']
            request.user.image = image
            request.user.save()
            return HttpResponseRedirect(reverse("i:info"))
        else:
            return render(request, )

    # 第二种方法，直接实例化,实例化直接保存
    # def post(self, request):
    #     image_form = UploadImageForm(request.POST, request.FILES, instance=request.user)
    #     if image_form.is_valid():
    #         image_form.save()
```

9. 关于在form表单里面直接调用model的字段使用

```
# 上传图片  使用forms.ModelForm
class UploadImageForm(forms.ModelForm):
    class Meta:
        model = UserInfo  #说明要引用的model是那个
        fields = ['image'] # 说明要使用的字段有那个
```

10. 在修改页面的其他的信息的时候 和修改头像的信息类似;
11. view.py 把用户显示后的 post方法加上

```python
    # 用户信息提交的逻辑
    def post(self, request):
        info_form = UploadInfoForm(request.POST, instance=request.user)
        if info_form.is_valid():
            info_form.save()
            return HttpResponseRedirect(reverse("i:info"))
        else:
            return render(request, 'my_info.html', {'info_form': info_form})
```

12. 关于邮箱的修改,和手机的修改 大家可以下来的时候自己做.
13. 邮箱和手机的修改可以使用js.
14. 需要给数据库添加16-20门课程.先给添加 2个第一大类,在里面分别添加2个大二分类, 然后在添加4-5条数据