django模板与视图

一. django的MTV架构

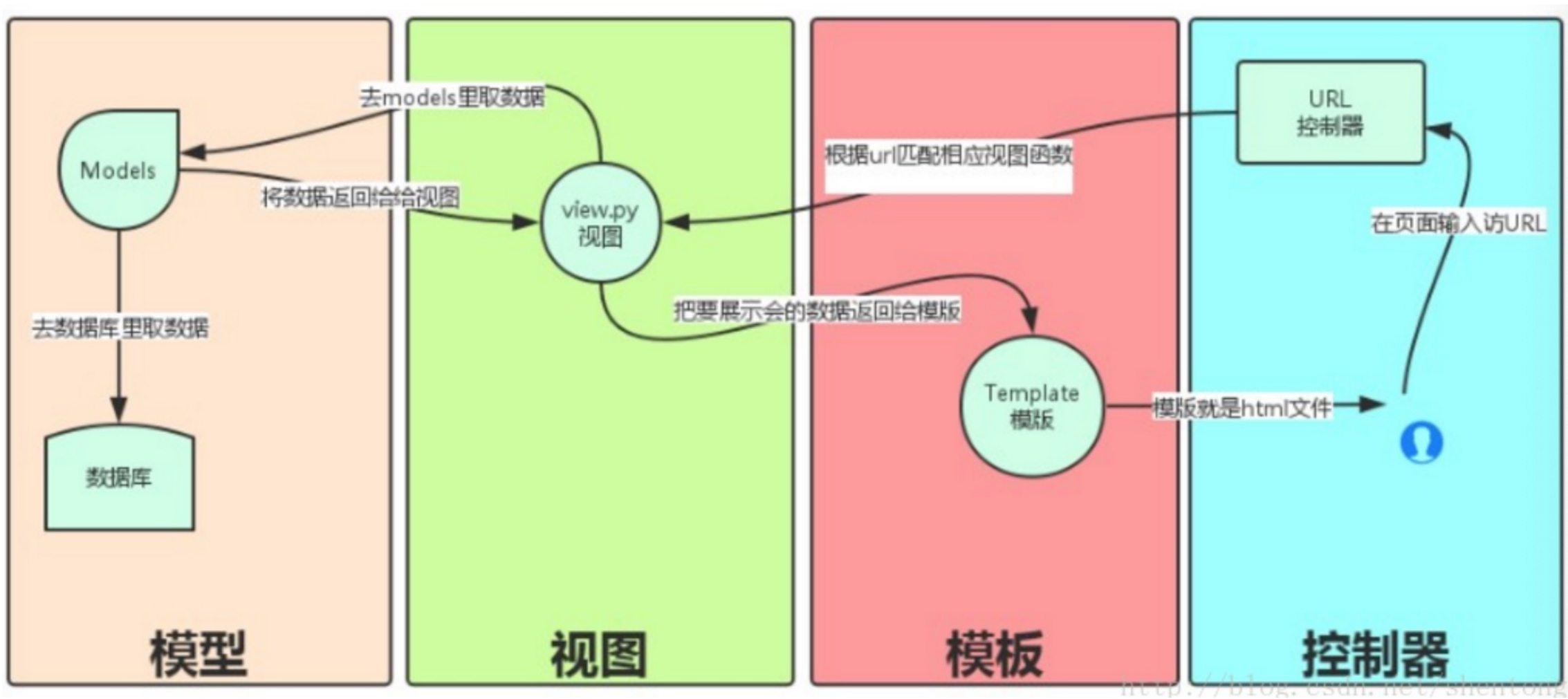
1.1 MVC是众所周知的模式, 即将应用程序分解成三个组成部分: model(模型),view(视图)和 controller(控制器).

M: 即数据存取层 该层处理与数据相关的所有事务.

V: 负责把数据格式化后呈现给用户, 即表现层. 该层处理与表现相关的任务, 如何在页面或其他类型文档中进行显示.

C: 接受外部用户的操作, 根据操作访问模型获取数据, 并调用"视图"显示这些数据. 即业务逻辑层.

在Django中通过模板(Templte)接受用户的参数以及将结果进行渲染以展示, 所以Django也称之为是一种MTV架构.



二. 第一个自建网站

2.1 创建APP

```
(webapps) [root@harbor-a webapps]#  django-admin startapp  login
```

2.2 将APP注册到项目:

只有将app注册到项目, 这样django才能找到app目录下的模板文件(app-name/templates/中的文件)与静态资源(app-name/static/中的文件)修改 setting.py 文件, 增加如下.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'login',
]
```

2.3 app视图函数编写

```
from django.shortcuts import render
from django.http import HttpResponse
def login(request):
    return HttpResponse("this is login page!")
```

2.4 视图函数与urls关联

2.4.1 Django2.0以上版本使用以下方法添加url映射

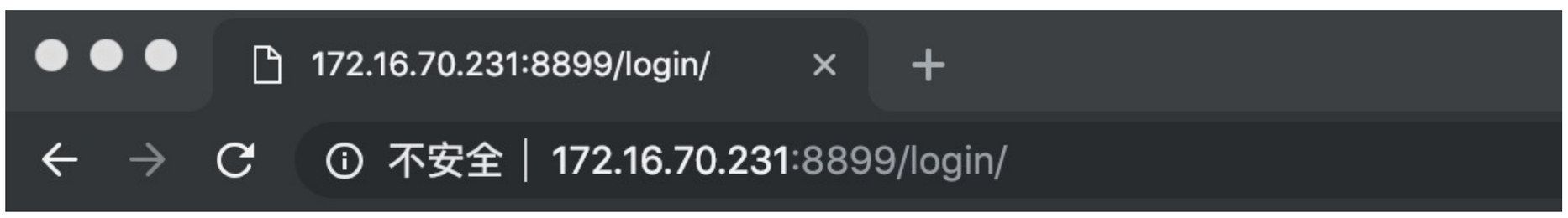
```
from login import views as login_views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('login/', login_views.login),
]
```

2.4.2 Django1.8 - 2.0版本使用以下方法添加url映射

```
from learn import views as learn_views  # new
urlpatterns = [
    url(r'^$', learn_views.index),  # new
    url(r'^admin/', admin.site.urls),
]
```

2.5 验证

启动项目后访问url, 可以看到如下页面.



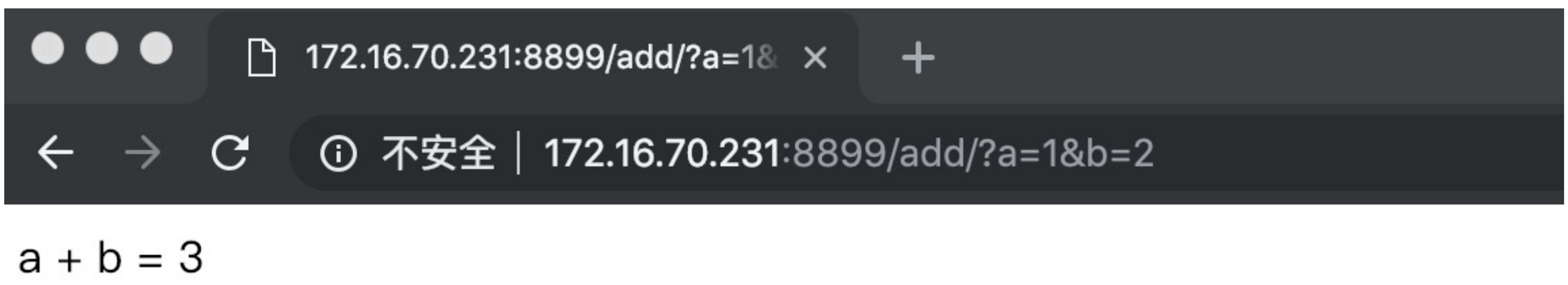
三. django接收http请求参数.

3.1 采用如下url的方式进行参数传递
http://172.16.70.130:8000/add?a=1&b=2 计算两数字相加

3.1.1 编写views函数
```
def add(request):
    a = int(request.GET.get('a'))
    b = int(request.GET.get('b'))
    return HttpResponse("a + b = %s" %(a+b))
```

3.1.2 定义urls映射
```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('login/', login_views.login),
    path('add/', login_views.add),  ##新增
]
```

3.1.3 验证:



3.2 采用如下url方式进行参数传递
http://172.16.70.130:8000/add/3/4/ 计算两个数字相加

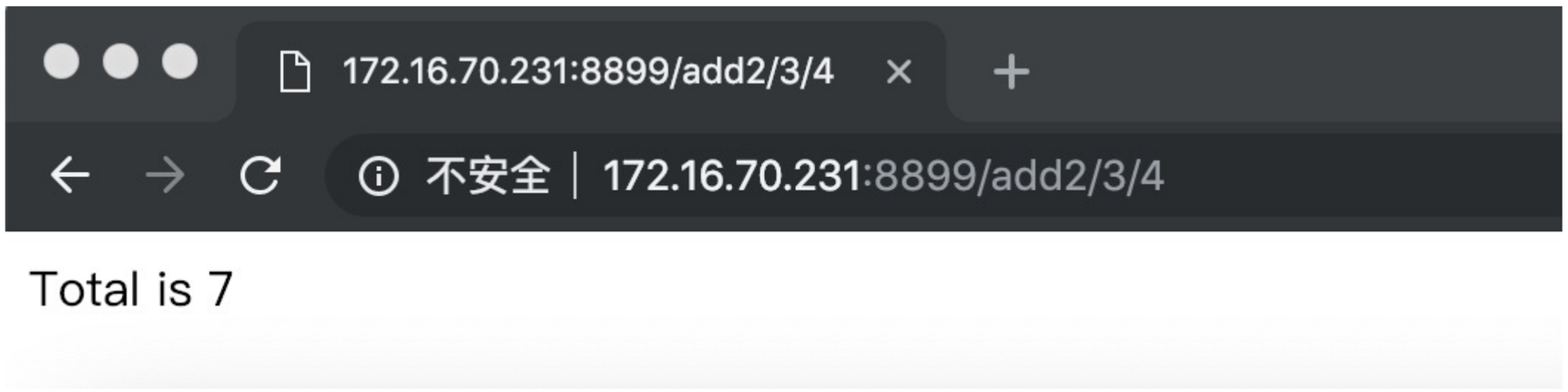
3.2.1 定义视图
```
def add2(request, a, b):
    c = int(a) + int(b)
    return HttpResponse("Total is %s" %c)
```

3.2.2 定义urls映射

Django2.0以上版本使用以下方式添加url映射
```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('login/', login_views.login),
    path('add/', login_views.add),
    path('add2/<int:a>/<int:b>', login_views.add2),
]
```

Django1.8 - 2.0版本使用以下方式添加url映射
```
    url(r'^add3/(\d+)/(\d+)/$', calc_views.add2),
```

3.2.3 验证:



3.3 django获取http请求参数:
以下均在视图函数中定义:
1. GET请求: request.GET.get("a")
2. POST请求: request.POST.get("a")
3. PUT请求:
```
   httpPut = QueryDict(request.body)
   a = httpPut.get("a")
```
4. DELETE请求:
```
   httpDel = QueryDict(request.body)
   a = httpDel.get("a")
```

四. django URL include详解
一般所熟知的就是在项目目录中的urls.py中的定义路由策略, 在urlpartterns列表中来配置url, 每一个列表项就是一个由url函数的调用. 但假如一个project中有多个app, 用以上的方式来管理url可能会造成比较混乱的局面, 为了解决这个问题, 我们可以用include的方法来配置url, 首先我们在自己的app中建立一个urls.py.

4.1 在app目录下新建urls.py文件并加入以下内容:
```
    from django.urls import path
    from login.views import login
    urlpatterns = [
        path('login/', login),
    ]
```

4.2 在项目目录下修改urls.py中的内容
```
    urlpatterns = [
        path('admin/', admin.site.urls),
        path('', include('login.urls')),
        ]
```

然后通过刚刚相同的url(http://172.16.70.231:8899/login)访问发现也可以访问了, 通过这样的url管理会发现更加整洁, 可扩展性更强.

五. django视图函数FBV与CBV

我们之前写过的都是基于函数的view, 就叫FBV, 还可以把view写成基于类的, 叫做CBV.

5.1 FBV示例代码:

```
    def login(request):
        return HttpResponse("this is login page!")
```

5.2 CBV示例代码:

```
    from django.views.generic import View
    class AddClass(View):
        def get(self, request):
            return HttpResponse('this is addclass get method')
```

```
        def post(self, request):
            pass
        def put(self, request):
            pass
        def delete(self, request):
            pass
```

5.2.1 CBV的urls映射关系的写法与FBV不同：

```
    urlpatterns = [
        path('admin/', admin.site.urls),
        path('', include('login.urls')),
        path('addclass/', login_views.AddClass.as_view()),
        ]
```