02 Django路由Router

一、路由router

在实际开发过程中，一个Django 项目会包含很多的 app ，这时候如果我们只在主路由里进行配置就会显得杂乱无章，所以通常会在每个 app 里，创建各自的 urls.py 路由模块，然后从根路由出发，将 app 所属的 url 请求，全部转发到相应的 urls.py 模块中。而这个从主路由转发到各个应用路由的过程叫做路由的分发

路由匹配

```
# 使用url给视图函数传参数
    path('index/', index)
    path('detail/<int:id>/', detail)

# 给url取别名，那么在使用此url的地方可以使用别名。比如：
    path('index/', index, name='index')
    path('detail/<int:id>/', detail,  name='detail')
```

命名空间

在实际应用中，Django中可能存在多个应用程序，每个应用程序都可能有自己的路由模块。为了防止路由冲突，Django提供了命名空间(namespace)的概念。命名空间是一种将路由命名为层次结构的方式，使得在查询路由时可以限定在该命名空间内。

```
# 在根路由中可以设置命名空间
  path('app/', include(('App.urls', "App"), namespace='App'))
```

反向解析

Django路由反向解析是一个非常重要的功能，它可以让我们在代码中使用路由别名替代URL路径，在修改URL时避免代码中的硬编码依赖，同时也可以提高可读性和可维护性。

```
# 在视图函数中，反向解析url：
    from django.shortcuts import render, redirect, reverse

    def buy(request):
        return redirect(reverse('index'))
        return redirect(reverse('detail', args=[2]))
        return redirect(reverse('detail', kwargs={"id": 2}))
```

```
# 在templates中，使用别名：
  {% url 'detail' stu.id %}

# 使用命名空间：
#  指定命令空间后，使用反向解析时需要加上命名空间，比如：
#   1. 在视图函数中：
       return redirect(reverse('App:index'))
#   2. 在templates中：
    {% url 'App:index' %}
```