上午课程核心内容

1. server 配置

```
server {
    # 指定 server 监听的 ip和port
    listen 80;
    # 指定 server 服务的虚拟域名
    server_name www.meiduo.site;

    # 指定客户端请求的文件去哪查找
    root /data/meiduo/front_page/;
    # 指定客户端请求时未指明请求文件时默认返回的文件
    index index.html;

    # location 地址配置
    location = / {
        try_files $uri $uri/ =404;
    }
}
```

2. location 配置

配置格式:

```
location [匹配规则] [匹配路径] {
  # 匹配成功执行的操作
}
```

匹配规则:

类型	含义	匹配方式	优先级	样式
=	精确匹配	前缀	1	location = /image {}
^~	优先匹配	前缀	2	location ^~ /page {}
~	普通正则：大小写敏感	正则符号	3	location ~ .(jpe?g)$ {}
~*	普通正则：大小写不敏感	正则符号	3	location ~* .(jpe?g)$ {}
空 /	通用匹配	前缀	4	location / {}
空 <路径>	前缀匹配	前缀	*	location /index {}

匹配优先级：

精确匹配＞location 完整路径＞优先匹配＞正则匹配＞location 部分路径＞通用匹配

匹配举例：

```
location = / {
   # 精确规则A
}
location = /login {
   # 精确规则B
}
location ^~ /static/ {
   # 优先规则C
}
location ~ \.(gif|jpg|png|js|css)$ {
     # 正则规则D
}
location ~* \.png$ {
     # 正则规则E
}
location / {
     # 通用规则F
}
```

```
访问 http://a.com/：规则A
访问 http://a.com/login：规则B
访问 http://a.com/static/a.html： 规则C
访问 http://a.com/b.png：规则D
访问 http://a.com/static/c.png： 规则C
访问 http://a.com/a.PNG：规则E
访问 http://a.com/category/id/1111：规则F
```

3. root 和 alias 的区别

location配置项下可以通过 root 或 alias 指定请求文件的查找目录

区别： 1）root：将匹配的 ur i路径和 root 路径直接拼接去查找文件.

2）alias：先将已匹配部分从 uri 路径中剔除，剩余部分和 alias 路径拼接去查找文件.

示例：

```
效果一：
location /txt/ {
    alias /var/www/txt/;
}
```

```
效果二：
location /txt/ {
    root /var/www/txt/;
}
```

结果：

```
效果一：访问http://localhost/txt/1.txt，nginx找/var/www/txt/1.txt文件
效果二：访问http://localhost/txt/1.txt，nginx找/var/www/txt/txt/1.txt文件
```

4. 美多商城前端页面 nginx 部署

```
server {
    listen 80;
    server_name www.meiduo.site;

    location / {
        root /data/meiduo/front_page/;
        index index.html;
        try_files $uri $uri/ =404;
    }

    location /static {
        alias /data/meiduo/front_page/;
    }
}
```

处理 /static 开头的地址，查找文件时先将 /static 部分从访问地址中去除，然后再去查找对应文件

5. 反向代理配置

正向代理和反向代理：

1）正向代理：客户端搭建的，帮助客户端去请求外部的网络，隐藏客户端的身份。

2）反向代理：服务端搭建的，帮助服务端去接收客户端的请求，隐藏服务端的身份。

美多商城部署-动态请求转发：

```
root@itcast:/etc/nginx/conf.d# vi meiduo.conf

server {
    listen 80;
    server_name www.meiduo.site;

    location = / {
        root /data/meiduo/front_page/;
        index index.html;
        try_files $uri $uri/ =404;
    }

    location /static {
        alias /data/meiduo/front_page/;
    }

    location ~ \.html$ {
        root /data/meiduo/front_page/;
    }

    location / {
        include uwsgi_params;
        uwsgi_pass 192.168.19.131:8001;
    }
}
```

这 3 个 locaiton 是用来处理静态页面请求的，
即 / 地址、static 开头的地址或 .html 结尾
的地址，这些地址都是在请求静态页面

nginx 收到客户端请求时，除上面 3 类地址之外的地址，
都属于动态请求地址，直接转发给对应的后端 uwsgi 服务处理

下午课程核心内容

1. 负载均衡配置

基本概念:

所谓的负载均衡，指的就是通过反向代理的配置，让代理服务器将收到的客户端请求动态分发到不同的后端服务器上面，以此来达到多个后端服务共同处理客户端请求的目的，减轻每一台后端服务器的压力，这个过程就叫负载均衡。

配置过程:

```
# backends 这个名字不固定
upstream backends  {
  server [域名|ip]:port;
  server [域名|ip]:port;
  server [域名|ip]:port;
  ...
}

server {
  location / {
    proxy_pass  http://backends;
    或
    include uwsgi_params;
    uwsgi_pass backends;
  }
}
```

美多商城负载均衡配置：

```
upstream meiduo {
    server 192.168.19.131:8001;
    server 192.168.19.131:8002;
}

server {
    listen 80;
    server_name www.meiduo.site;

    location = / {
        root /data/meiduo/front_page/;
        index index.html;
        try_files $uri $uri/ =404;
    }

    location /static {
        alias /data/meiduo/front_page/;
    }

    location ~ \.html$ {
        root /data/meiduo/front_page/;
    }

    location / {
        include uwsgi_params;
        # uwsgi_pass 192.168.19.131:8001;
        uwsgi_pass meiduo;
    }
}
```

upstream 中指定所有后端服务的地址

配置请求转发实现负载均衡

负载均衡的常见算法：

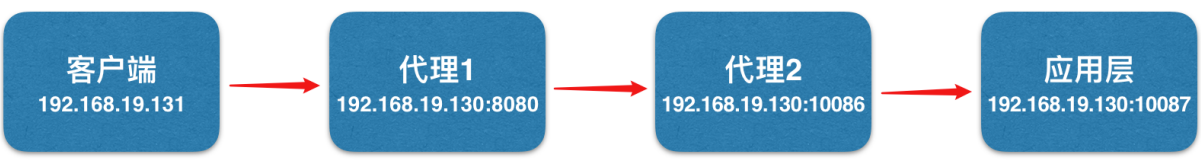
轮询(默认)：请求按顺序逐一分配到不同的后端服务器。
weight：指定轮询权重，值越大，分配到的几率就越高，适用于后端服务器性能不均衡情况。
ip_hash：按访问IP的哈希结果分配请求，分配后客户端访问固定后端服务器，有效的解决动态网页会话共享问题。
fair：基于后端服务器的响应时间来分配请求，响应时间短的优先分配。
url_hash：按访问URL的哈希结果分配请求，使同一URL定向到同一台后端服务器，可提高后端缓存服务器的效率。

2. Nginx 日志配置

日志文件：

日志文件	说明
access.log	可以得到用户请求的相关信息
error.log	可以获取某个 web 服务故障或其性能瓶颈等信息

配置案例：



需求：10087 服务在输出日志时，通过 $remote_addr 获取真实客户端的 IP 地址。

配置过程：

```
① 每一层：proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
② 应用层(指要输出日志的那个server)：
  real_ip_header X-Forwarded-For;
  # 如：set_real_ip_from 192.168.0.0/16;
  set_real_ip_from IP样式/匹配位数;
  real_ip_recursive on;
```

3. Docker 在部署时的作用

可以快速搭建项目运行所依赖的环境。

mysql、redis、fastdfs、es 等绝大部分部署需要的环境，都可以使用 Docker 快速搭建运行，而不需要按照传统的软件安装之后，配置再进行运行。

4. Docker 数据管理

目的：**容器中产生数据的持久化保存**.

方式：

数据卷和数据卷容器：

本质：都是将宿主机的一个目录或文件和容器中的目录或文件进行映射，映射之后，容器中保存数据时，其实就是保存到宿主机中，这样即使容器被删除，数据因为保存在宿主机，所以也不会丢失。

宿主机：你的的容器是在哪个机器创建的，哪个就是我们所说的宿主机。

5. Docker 网络管理

目的：**容器内部能够使用网络进行通信**.

方式：

端口映射和设置 host 网络模式。

端口映射：

```
-P：随机端口映射
-p <宿主机ip>:<宿主机port>:<容器port>：指定端口映射
```

设置 host 网络模式：

> 注：设置容器为 host 网络模式时，宿主机中对应的端口不能被占用，否则容器启动会失败

```
docker run --network=host -itd --name <容器名称> <镜像名称>
```