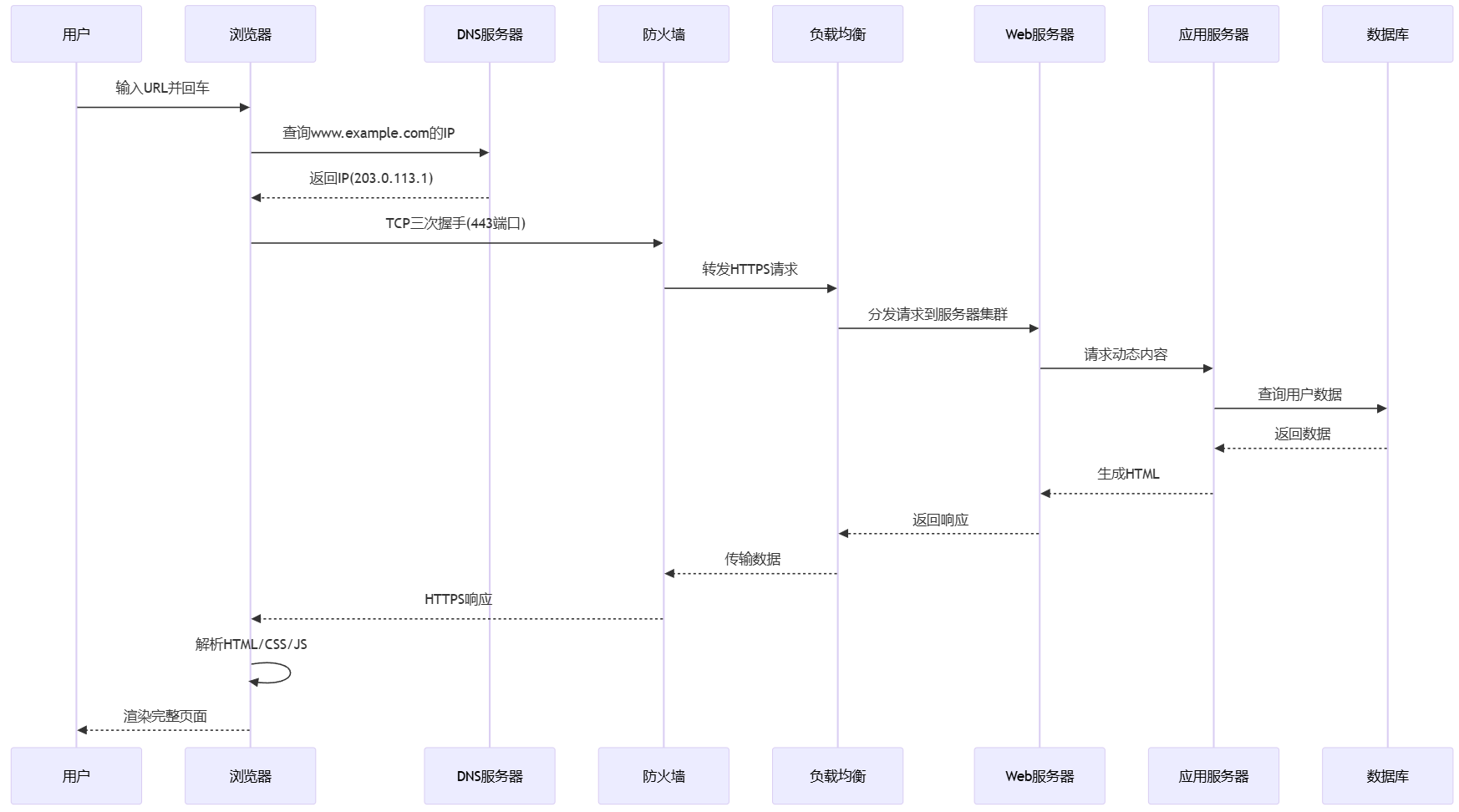
# 安全基础

## 网站工作原理

### 网站访问完整过程



**1. DNS解析：**

* 浏览器缓存 → 系统hosts → 本地DNS → 根DNS → 顶级域DNS
* 使用dig +trace www.example.com查看完整过程

**2. TCP连接：**

* 三次握手：SYN → SYN-ACK → ACK
* HTTPS增加TLS握手（ClientHello → ServerHello → 密钥交换）

**3. 内容获取：**

* 静态资源：直接由Web服务器(Nginx/Apache)返回
* 动态内容：通过PHP/Python等应用服务器生成

**4. 浏览器渲染：**

* 解析HTML构建DOM树
* 解析CSS构建CSSOM树
* 合并成渲染树 → 布局 → 绘制

## 浏览器核心工作原理

浏览器架构组件：

表9.1 浏览器架构组件

|  |  |
| --- | --- |
| **组件** | **功能** |
| 用户界面 | 地址栏/书签/后退按钮等交互元素 |
| 浏览器引擎 | 协调UI与渲染引擎工作 |
| 渲染引擎 | 解析HTML/CSS并显示内容(Blink/WebKit) |
| JS解释器 | 执行JavaScript代码(V8/SpiderMonkey) |
| 网络模块 | HTTP/FTP等网络请求(Chromium使用Cronet) |
| 数据存储 | Cookie/IndexedDB/LocalStorage |

**1. 关键路径优化：**

* 优化CSS：减少选择器复杂度，避免阻塞渲染
* JS异步加载：<script async>或<script defer>
* 资源预加载：<link rel="preload">

**2. 现代浏览器特性：**

* 渐进式渲染：分块显示已下载内容
* 合成层加速：GPU渲染CSS动画
* Service Worker：离线缓存支持

## 网站安全威胁全景图

客户端威胁：

表9.2 网站安全之客户端威胁

|  |  |  |
| --- | --- | --- |
| **威胁类型** | **原理** | **防护措施** |
| XSS(跨站脚本) | 恶意脚本注入页面执行 | 输入过滤/输出编码/CSP策略 |
| CSRF(跨站请求伪造) | 诱导用户发起非预期操作 | Anti-CSRF Token/同源检测 |
| 点击劫持 | 透明层覆盖诱导点击 | X-Frame-Options响应头 |
| 中间人攻击 | 拦截通信数据 | HTTPS/HSTS/证书锁定 |

服务端威胁：

表9.3 网站安全之服务器威胁

|  |  |  |
| --- | --- | --- |
| **威胁类型** | **原理** | **防护措施** |
| SQL注入 | 恶意SQL篡改数据库查询 | 参数化查询/ORM框架 |
| 文件上传漏洞 | 上传恶意文件获取控制权 | 文件类型校验/隔离存储 |
| 命令注入 | 通过输入执行系统命令 | 输入过滤/最小权限原则 |
| DDoS攻击 | 海量请求耗尽服务器资源 | WAF/CDN清洗/流量限速 |

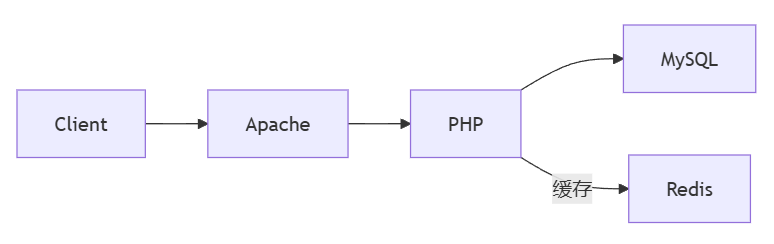
## 静态网站 vs 动态网站

表9.4 静态网站 vs 动态网站

|  |  |  |
| --- | --- | --- |
| **特性** | **静态网站** | **动态网站** |
| **内容生成** | 预先生成HTML文件 | 实时生成HTML |
| **技术栈** | HTML/CSS/JS | PHP/Python/Ruby + 数据库 |
| **服务器负载** | 低(仅文件传输) | 高(需执行计算) |
| **典型场景** | 企业官网/博客/文档 | 电商/社交平台/管理系统 |
| **更新方式** | 修改文件重新上传 | 后台管理系统更新 |
| **CDN支持** | 完美支持 | 仅静态资源可缓存 |
| **代表技术** | Hugo/Jekyll/Gatsby | WordPress/Django/Spring Boot |

## 主流WEB服务架构

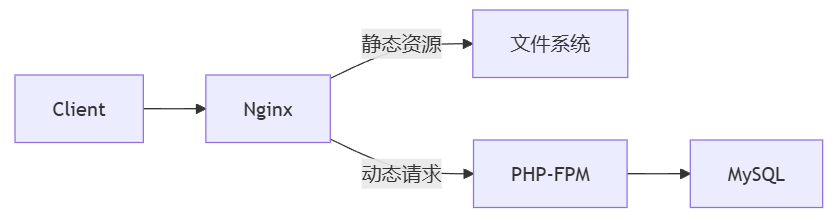
**1. LAMP 架构（经典Linux方案）**



**组件作用：**

* Linux：操作系统基础
* Apache：HTTP请求处理/静态资源服务
* MySQL：关系型数据存储
* PHP：动态内容生成

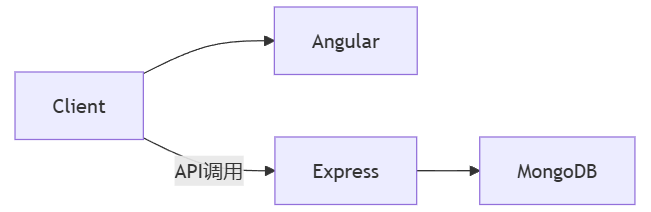
**2. LNMP 架构（高性能方案）**



**性能优化点：**

* Nginx：事件驱动架构，高并发处理
* PHP-FPM：进程管理器，资源隔离
* Opcache：PHP字节码缓存

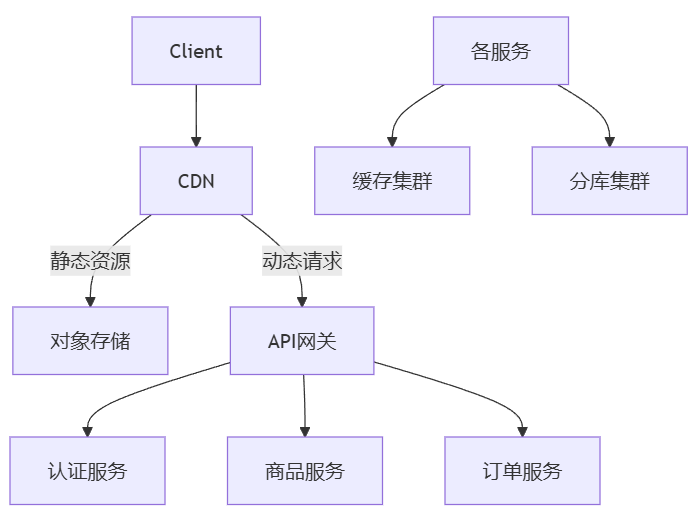
**3. MEAN 架构（全JavaScript方案）**



**组件作用：**

* MongoDB：JSON文档数据库
* Express：Node.js后端框架
* Angular：前端框架
* Node.js：JavaScript运行时

**4. 云原生架构（微服务方案）**



**核心组件：**

* Docker/K8s：容器化与编排
* Service Mesh：服务通信治理
* ELK：日志监控系统

## 拓展

**安全最佳实践：**

* 所有传输启用HTTPS（使用Let's Encrypt免费证书）
* 实施CSP(内容安全策略)防止XSS
* 定期进行渗透测试（工具：OWASP ZAP/Burp Suite）
* 遵循最小权限原则部署服务