第五章——设计工程

软件架构设计

部署视图（物理视图）

部署风格：

client/server

Client*Server*是建立在局域网的基础上的.*Browser*Server是建立在广域网的基础上的。

C/S架构的优缺点：

*优点：

1.客户端因为是独立设计，所以可以实现个性化

2.因为客户端是需要进行安装的，可以不需要重复安装和加载

3.因为客户端是独立开发的，所以有能力对客户端进行安全设计

4.如果遇到不同的操作系统，需要为不同的操作系统各开发一套客户端

*缺点：

1.因为客户端是不需要重复安装，所以用户可以不更新与升级，增加了维护成本。

2.因为需要开发客户端和服务器两套程序，所以开发成本会增加

B/S架构的优缺点：

*优点：

1.因为B/S架构具备通用性，所以开发成本较低。

2.因为不需要安装客户端，所以客户端不需要进行升级，只需要更新后台代码即可实现所有客户端的更新。

3.因为B/S架构多用WEB网页进行开发，所以增、删功能也非常容易，只需要修改网页即可完成

*缺点：

1.耗流量，每次都要加载全部的内容（不过有缓存可以降低流量损耗）

2.因为没有独立的客户端，所以无法实现个性化（通过账号体系可以实现）

3.因为没有独立设计客户端，所以客户端难以实现安全控制（HTTPS、控件）。

4.难以实现特殊的操作（删本地文件），所以所有的杀毒软件都是C/S架构的。

B*S*架构更多的时候是使用了*HTTP*协议、而*C*S架构更多的时候使用的WinSocket协议（TCP、UDP）

P2P物理架构：
所有的计算机节点都是对等的
3-tier
fat-client
fat-server
Distributed Client/server

逻辑视图
把软件划分成多个模块，重点考虑功能、可维护性

所有软件都需要定义逻辑架构
逻辑架构风格
表现层分离风格：MVC
数据流风格（Dataflow）：批处理序列、管道––过滤器风格
调用/返回风格：主程序/子程序、面向对象（ADT）
分布计算风格：多层（layer）、代理、C/S 、P2P
独立构件风格
虚拟机风格
层次架构（layered architecture）
application––business––Middleware––system software

中间件：
1.数据访问中间件

2.消息中间件

3.远程过程调用RPC中间件

4.事物中间件

5.分布对象中间件

6.web服务中间件

MVC
!MVC视图

Model：管理系统中存储的数据和业务规则，并执行相应的计算功能

View：根据模型生成提供给用户交互界面，不同的视图可以对相同的数据产生不同的界面

Control： 接收用户的输入，通过调用模型获得响应，通知视图更新

3 Tiers
!3Tier视图

表示层：负责向用户呈现界面，接收用户请求发送给业务逻辑层

业务逻辑层：负责执行业务逻辑处理用户请求，并调用数据访问层提供的永久性操作

数据访问层：负责执行数据持久型操作

管道和过滤器
!管道和过滤器
.
每个组件都执行data的输入和输出

服务与微服务的架构风格
微服务架构风格使用小服务来开发单个应用

仓库风格
以数据为中心 分为

数据库系统：以数据库为核心

超文本系统：超链接将文本图片组织在一起

黑板系统：为参与问题解决的知识源提供了共享的数据表示，这些数据表示是与应用相关的。在黑板系统中，控制流是由黑板数据的状态决定的，而非按照某个固定的顺序执行。
微内核风格
微内核概念来源于操作系统，占用很小的空间向用户提供标准接口，使用户能够模块化扩展功

能。

进程与线程

- 进程是一个内存中的可执行程序, 提供程序运行的各种资源。

- 线程是系统处理机调度的基本单位.一个进程拥有多个线程，
  线程可以执行进程中的任意代码, 进程中的所有线程共享进程
  的虚拟地址空间和系统资源。

- 在单核时代实现并发处理，采取分时的方式，即把CPU的时间
  分成很多片段，某个时间片只能执行某个线程。多核cpu则实 现了并行处理。

构件图

构件(Component)
代表系统的一个物理实现模块，代表逻辑模型元素如类 、接口、协同等的物理打包
接口(Interface)
由一组操作组成，它指定了一个契约，这个契约必须由 实现和使用这个接口的构件的所遵循。

接口分提供接口和请求接口
端口(Port)
描述了在构件与它的环境之间以及在构件与它的内部构 件之间的一个显式的交互点